PROTEIN-FUNCTION PREDICTION USING A GRAPH-MATCHING ALGORITHM

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By
Sarjan Shrestha

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

August 2014

Fargo, North Dakota
Title

PROTEIN FUNCTION PREDICTION USING GRAPH MATCHING ALGORITHM

By

Sarjan Shrestha

The Supervisory Committee certifies that this disquisition complies with North Dakota State University’s regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Changhui Yan
Chair

Dr. Jun Kong

Dr. Limin Zhang

Approved:

09/19/2014
Date

Dr. Brain M. Slator
Department Chair
ABSTRACT

Proteins are the basic building blocks for biological life. Proteins interact with other proteins or molecules to carry out various functions. The active region in the protein molecule is the functional site. Today, most research work has been done to establish sequence-based, structure-based, and machine-learning approaches for prediction of protein’s functional site. This paper describes a tool for identifying the functional site of protein using a graph-based approach. Each protein structure is represented as a graph. Each protein residue represents a vertex, and the relationship between residues represents the edges of the graph. The tool compares proteins to identify the functional site in the form of maximum common subgraphs using McGregor’s subgraph algorithm. Functional sites for new proteins can be predicted using the subgraphs. The graph-based approach is computationally efficient and accurate due to the wide range of protein properties that are taken into consideration.
ACKNOWLEDGMENTS

I greatly appreciate my adviser, Dr. Changhui Yan, for his continuous motivation and guidance for my research. He has bestowed his time to provide me with deeper insight for the successful completion of my project. I am very thankful for my committee members, Dr. Jun Kong and Dr. Limin Zhang, and for their support and time.

I am also thankful for Stephanie Sculthorp. She provided notification about the dates, deadlines, and information regarding the Computer Science department’s policies.
TABLE OF CONTENTS

ABSTRACT .............................................................................................................. iii

ACKNOWLEDGMENTS ............................................................................................. iv

LIST OF TABLES ..................................................................................................... vi

LIST OF FIGURES ................................................................................................ vii

CHAPTER 1. INTRODUCTION .................................................................................. 1

1.1. Protein Function ............................................................................................. 1

1.2. Protein Structure Representation ................................................................. 2

1.3. Protein Functional Sites ............................................................................... 4

1.4. Properties of Protein and Peptide Bond ...................................................... 4

1.5. Prediction of Protein Functional Sites ......................................................... 6

CHAPTER 2. APPROACH BASED ON MAXIMUM COMMON SUBGRAPH .............. 10

2.1. Maximum Common Subgraph ..................................................................... 10

2.1.1. Maximum Common Subgraph Algorithm ............................................. 11

2.1.1.1. The McGregor Algorithm ................................................................. 11

2.1.1.2. The Durand-Pasari Algorithm .......................................................... 12

2.1.1.3. The Balas-Yu Algorithm ................................................................. 12

2.2. Tool Implementation using McGregor Algorithm ...................................... 13

2.3. Overview of the Tool ................................................................................... 17

CHAPTER 3. DISCUSSION AND CONCLUSION .................................................. 21

BIBLIOGRAPHY ...................................................................................................... 26
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Major classes and methods that are in implemented tool</td>
<td>17</td>
</tr>
<tr>
<td>2. The result of a comparison for proteins with the number of common residues found</td>
<td>21</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Protein Structure</td>
<td>3</td>
</tr>
<tr>
<td>2. The McGregor Algorithm</td>
<td>11</td>
</tr>
<tr>
<td>3. The Durand-Pasari Algorithm</td>
<td>12</td>
</tr>
<tr>
<td>4. The Balas-Yu Algorithm</td>
<td>13</td>
</tr>
<tr>
<td>5. Protein Structure Representation in .fog Format</td>
<td>15</td>
</tr>
<tr>
<td>6. Command for file_conversion</td>
<td>18</td>
</tr>
<tr>
<td>7. Command for subgraph_finding</td>
<td>19</td>
</tr>
<tr>
<td>8. (a) Sample Graph 1 with .fog Representation, (b) Sample Graph 2 with .fog Representation, and (c) Output Sample Common Subgraph Graph with .fog Representation</td>
<td>20</td>
</tr>
<tr>
<td>9. Screenshot for file_conversion Command</td>
<td>22</td>
</tr>
<tr>
<td>10. Screenshot for subgraph_finding Command</td>
<td>22</td>
</tr>
<tr>
<td>11. Protein Structure Diagram for 1floA</td>
<td>23</td>
</tr>
<tr>
<td>12. Protein Structure Diagram for 1ignA</td>
<td>23</td>
</tr>
<tr>
<td>13. Common Residues Between Proteins: 1floA and 1ignA</td>
<td>24</td>
</tr>
</tbody>
</table>
CHAPTER 1. INTRODUCTION

Advancement in the field of bioinformatics has led to efficient approaches to identify the protein’s functional sites using a computational method. With the availability of high-speed computers, it is possible to analyze the vast data available in the protein genome in order to identify or predict new functional sites. Functional site identification contributes to the development of new drugs and development of synthetic biochemistries. Computational approaches to predict a functional site include a similarity search, a protein structure comparison, and a machine-learning approach. The tool presented in this paper provides an easy interface for graph-based similarity searching between proteins where the proteins are represented as a graph. The following sections in the chapter give a brief introduction for the Protein Function, the protein’s structural representation, the concept and significance of protein’s functional site, and an overview of existing approaches to predict the protein’s functional site. The second chapter describes a graph-based approach to predict the functional site. Different algorithms to identify the maximum common subgraph in graphs are also presented. In the third chapter, the result of the comparison between various proteins using the subgraph finding tool is presented.

1.1. Protein Function

Proteins consist of amino-acid chains joined together by peptide bonds. Proteins have various functionalities that include replicating DNA, responding to stimuli, and catalyzing chemical reactions [1]. Amino-acid sequences are determined by the information encoded in the genes. These sequences of amino acid, in turn, determine the protein’s structure. This structure determines the functional characteristics. Changes in protein structure can occur due to the influence of other molecules to which it binds or under different environmental effects. These
changes in the structure of protein are responsible for the diversity in the function protein exhibits. Therefore, understanding the protein structure is very important to obtain a clear idea about the protein’s function. This knowledge is very important because it provides important clues to understand various biochemical reactions in biological life. Protein’s function knowledge also helps to predict the protein’s binding site that plays an important role in the design of drugs. A drug targets the binding site of proteins to enhance the desirable functionality or inhibit the delirious functionality of the protein. A protein interacts with other proteins to form a protein complex. Protein-protein interactions have different biological functions, such as gene expression, enzymatic reactions, signal transduction, inter-cellular communications, and immunoreactions [2]. Various research studies have been done for protein function prediction in the case of individual protein and protein complexes.

1.2. Protein Structure Representation

Protein can be described using four levels of structure representations: primary, secondary, tertiary, and quaternary [1]. Figure 1 shows the visual representation of the different structure.

i) Primary Structure: Primary structure refers to the linear sequence of amino acids held by peptide bonds which are formed during the process of protein biosynthesis or translation [3].

ii) Secondary Structure: Secondary structure is formed by hydrogen-bonding interactions between the amine group and the carboxylic group in the invariant parts of the amino acids. Secondary structures can take form of α-helices or β-sheets.

iii) Tertiary Structure: Tertiary structure is the overall shape of a single protein molecule. Protein in the tertiary structure is in the maximum stable state or the lowest energy state formed
by stabilizing forces due to the bonding interactions between side-chain groups of amino acids.
The tertiary structure is responsible for or controls the basic functionality of protein [4].

iv) Quaternary Structure: Quaternary structure is the organization of protein subunits
which function as a single protein complex. The structure of the protein complex is formed by a
stabilizing force due to various interactions among the different protein subunits [5].

Figure 1. Protein Structure [6]
1.3. **Protein Functional Sites**

Protein functions include different types of activities that occur in biological life when proteins interact with other proteins or molecules. The role of a protein includes the catalysis of biochemical reactions and the formation of a complex network that interact with other proteins to perform complex physiological functions, such as metabolic pathway operation and signal transduction [4]. Functional sites are the region that determines the activity and interactions among proteins or molecules [7]. Functional sites are, therefore, the point of interest for developing drugs because the sites act as the target that interacts to exert a physiological function.

Assigning a protein function to the uncharacterized protein by comparing it with a characterized protein based on the sequence or structure similarity has become an important topic for research in the field of molecular biology. The process of transferring the function name from a known protein to an unknown protein based on similarity is termed the protein-function prediction [4]. Similarity is measured by comparing a known protein with an unknown one. Comparison varies with different representations for the proteins. For instance, similarity can be measured using a sequence of amino acids or the spatial location of the amino acids.

1.4. **Properties of the Protein and Peptide Bond**

The side chains of a protein’s amino acids have different chemical properties. Different side chains cause differences for the tendency to participate in interactions with other amino acids and water. These variations influence the protein’s overall stability and function.

Amino acids can be classified as hydrophobic, hydrophilic and amphipathic based on the interaction with water. Hydrophobic amino acids are non-polar and uncharged; they have a
tendency to avoid contact with water. Hydrophobic amino-acid residues only engage in van der Waals interactions. Hydrophilic amino-acids are polar and charged; they have a tendency to interact with water. In polymer chains hydrophilic side chains tend to associate with other hydrophilic side chains or water molecules by means of hydrogen bonds. Amphipathic amino acids have both polar and non-polar characteristics as well as a tendency to form interfaces between hydrophobic and hydrophilic molecules. Amino acids are covalently joined by peptide bonds. The peptide bond’s properties have an important effect on the stability and flexibility of polypeptide chains in water. However, with folded protein, most stabilization energy is from non-covalent interactions. The non-covalent interactions are caused by the electrostatic attraction between opposite charges. Significant non-covalent interactions for a folded protein are electrostatic interaction, van der Waals interaction and the hydrogen bond. Van der Waals interactions are due to the weak attractive force between two atoms or groups of atoms, that results from the fluctuations in the electronic distribution around the nuclei. A hydrogen bond is an interaction between the donor atom, which is bound to a positively polarized hydrogen atom, and the acceptor atom, which is negatively polarized. The strength of non-covalent interactions depends, to some extent, on the environment. The ability of water molecules to form a hydrogen bond with the polar group has important effects on the energetic consequences for the proteins’ folding and stability. Considering these properties during the protein-function prediction leads to accurate results [8].
1.5. Prediction of Protein Functional Sites

A protein’s functional-site prediction requires a protein sequence or structural data to analyze. Protein data are obtained using experimental methods. Popular experimental methods include: mass spectrometry for sequence identification [9] and x-ray crystallography for structure identification [10]. Experimentally generated data are, hence, utilized for the analysis to determine the protein’s functional site.

The Protein Data Bank (PDB) is a publicly accessible database that stores three-dimensional structure representations of proteins that are experimentally determined. The stored data consist of information about atomic coordinates, molecule names, primary and secondary structure information, and ligand and biological assembly information [1].

Protein-function prediction can be done on a small scale using experimental approaches. With these methods, a single protein or a small set of proteins is targeted. The experimental-based protein-function identification includes gene knockout, targeted mutation, and the inhibition of gene expression [5]. Experimental-based approaches require a lot of human effort and time to identify the protein’s function. To address the proteins’ functional-site prediction in large scale, computational approaches are used. These methods utilize different computational algorithms to predict the functional site. With the available high-computation power, the computational-based approach can address the protein-function identification on a large scale. Computation approaches include sequence-based, structure-based, machine-learning and graph-based approaches.

The sequence-alignment and comparison tool BLAST( Basic Local Alignment Search Tool) [11] is based on finding sequence similarity between known and unknown proteins.
BLAST searches standard databases to find proteins that are homologous to the source protein. BLAST helps to predict the function of unclassified proteins based on the similarity score obtained by comparing an unclassified protein sequence with the sequence for a known protein. The sequence-based approach is inefficient for a large-sequence comparison due to a lot of needed computational time. Also, the protein function for a given protein sequence can vary during the evolution process when subjected to selective pressure [4]. Protein sequences have a stronger correlation with protein structure than protein function, so a protein sequence-based approach is not effective to predict protein function [12].

An alternative approach for protein-function prediction is based on the sequence-to-structure-to-function paradigm. With this approach, the structure for the protein of interest is determined. The protein structure provides insight for the function. Structure-based function identification is based on the fact that active protein sites, which are responsible for biochemical activity, are more conserved than the overall structure. The protein active-site comparison for the protein provides knowledge about identifying proteins that have the same function with the same global structure or possibly an unrelated global structure of protein [13]. Structural similarity can be determined by measuring the Euclidean distance between the corresponding protein residues. The structural-based method for identifying the area of similarity mentioned in [14] calculates the significance of root mean squared deviation (RMSD) between protein residues in three dimension space. The method uses geometric models to estimate the significance a priori. The significance measure helps to differentiate between the true functionally significant patterns and the patterns that occur by chance.
The machine-learning approaches for function prediction are based on constructing classification models. These models are created using the training data set available for proteins. The different machine-learning approaches to predict protein function are based on the classification model that is used. Machine-learning approaches include Support Vector Machine (SVM), k-nearest neighbor, and neural network. SVM classification is based on an analysis of the protein sequences' physicochemical properties [15]. The SVM classifier is trained using a sample of proteins in the functional class and proteins not in that class with different features taken into consideration. Every protein is represented by a feature vector with an encoded representation of tabulated residue properties, such as amino-acid composition, hydrophobicity, polarity, charge, and surface tension in the form of three descriptors, composition (C), transition (T) and distribution (D). The neural-network based enzyme active sites prediction from a three dimensional protein structure is presented in [16]. The residue score for the protein structure is determined by the likelihood of being catalytic. A neural-network is trained based on the experimentally validated dataset of proteins for which the catalytic residues have been located. Structural parameters, such as solvent accessibility, type of secondary structure, conservation score, and residue type, are used as neural-network inputs. The accuracy of the machine-learning approaches depends on the properties that are considered to construct the classification models. Not all of the properties can be considered to construct classification models for the machine-learning approach.

With the graph-based approach for protein function prediction, proteins are represented in the form of graphs. A graph representation for proteins helps to apply different graph-mining algorithms to identify the common recurring structures among various proteins [17]. The tool
mentioned in this paper utilizes the graph-based approach for protein-function identification. Amino-acid residues in the protein are represented as graph vertices, and the relationship between amino-acid residues in the protein sequence is represented as edges. Different properties of amino-acid residues are added as label for vertex and relationship properties are added as edge of graph.
CHAPTER 2. APPROACH BASED ON THE MAXIMUM COMMON SUBGRAPH

Graphs can be used to represent complex networks and show the network components. Graphs can store network’s information in a data structure that can be used efficiently for mathematical computations. Graph structure matching involves utilizing the subgraph isomorphism algorithm to identify the maximum common subgraph. With proteins, graph structure matching plays an important role in determining the evolutionary relationship and functional commonalities. Using a computational method based on graph structure matching, common subgraphs can be identified among the protein structures. Those subgraphs provide insight about the important functional sites.

2.1. Maximum Common Subgraph

Graph G is a 4-tuple G=(V,E,α,β), where V is the set of vertices, E is the set of edges, α: V → P is the properties associated with vertices, and β: E → P is the properties associated with edges [4]. Assuming, G=(V,E,α,β), G’=(V’,E’,α’,β’), and G”=(V’’,E’’,α’’,β’’) are three graphs, graph G”=(V’’,E’’,α’’,β’’) can be considered as the subgraph of graph G, if V’’ is subset of V and α(v)=α’’(v’’) for all v ϵ V ,v’’ ϵ V’’, E’’ is a subset of E; and β(e) = β’’(e) for all e ϵ E’’. A graph isomorphism between two graphs, G and G’ is the bijective mapping f: V → V’ with α(v) = α(f(v)) for all v ϵ V, and for every edge e(u,v) ϵ E there exists e’(f(u),f(v)) ϵ E’ such that β(e) = β’(e’). If f: V → V’ is a graph isomorphism between graph G and the subgraph of another graph, G’, then f can be defined as the subgraph isomorphism from G to G’. The common subgraph between graphs G=(V,E,α,β) and G’=(V’,E’,α’,β’), if there exists G’’ such that the subgraph isomorphism from G’’ to G and G’’ to G’ exists and |V’’|(|E’’|), is the number of common vertices(edges) in G and G’ for each subgraph isomorphism condition that is satisfied.
2.1.1. Maximum Common Subgraph Algorithm

Three different algorithms for finding the maximum common subgraph are mentioned in [4]. The algorithms are based on a depth-first search where which undesired search paths are pruned based on some heuristic. The heuristic used for pruning and the state of the utilized search space differentiate among these algorithms.

2.1.1.1. The McGregor Algorithm

The McGregor algorithm identifies a common subgraph through state “s”, initially an empty state, which represents common subgraphs under construction [4,7]. In each step of the graph searching, a pair of nodes, the first node from the first graph and the next one from the second graph, that is not explored is analyzed to deduce the possibility of extending the existing common subgraph state to the next state to add the pair of nodes to the existing state. In each state, if extension is possible with the new state being larger than the current one, the new state is generated as a branch for the search tree, and analysis is started for the new state. The search for the branch is pruned whenever the leaf is not enough to construct a subgraph that is larger than the current state. The algorithm is shown in Figure 2.

```
procedure McGregor_MCS(s)
while (NextPair(s,n1,n2))
    if (IsFeasiblePair(s,n1,n2)) then
        s' = AddPair(s,n1,n2);
        if (size(s') > CurrentMCSSize) then
            SaveCurrentMCS(s'); CurrentMCSSize = size(s');
            McGregor_MCS(s');
        end if
    end if
    if (!LeafOfSearchTree(s') and !PruningCondition(s')) then
        McGregor_MCS(s');
    end if
end while
end procedure
```

Figure 2. The McGregor Algorithm [4]
2.1.1.2. The Durand-Pasari Algorithm

The Durand-Pasari algorithm is based on pruning the search paths for a common subgraph of the maximum clique found in the graph [4,11]. Initially, an association graph is constructed with vertices that correspond to a pair of vertices for two starting graphs with the same label; the edges between the corresponding vertices are compatible with the pair of vertices being included. At each forward search, the new node is explored to determine if it is a legal node, i.e., a node connected in the current clique. If the clique size after adding the node is larger than the current largest clique without adding a node, the new clique is stored. The algorithm is shown in Figure 3.

```
procedure DurandPasari_MC(s)
    while (NextNode(s,n))
        if (IsLegalNode(s,n) && !PruningCondition(s)) then
            s' = AddNode(s,n);
            if (size(s')>CurrentMCSize) then
                SaveCurrentMC(s'); CurrentMCSize = size(s');
            end if
            if(!LeafOfSearchTree(s')) then DurandPasari_MC(s');
            end if
            BackTrack(s');
        end if
    end while
end procedure
```

Figure 3. The Durand-Pasari Algorithm [4]

2.1.1.3. The Balas-Yu Algorithm

The Balas-Yu algorithm is based on transforming the maximum common subgraph problem to the maximum clique detection with the heuristic being based on graph coloring and graph triangulation [4,14]. Each state “s” for the search paths is associated with a sub-problem to identify the maximum clique in a subgraph of the starting graph. Each sub-problem is characterized with three sets of parameters: included nodes, I; excluded nodes, Ex; and unclassified nodes, S. Initially, “n” nodes are chosen for set “S” and finding the number of nodes that can be colored in the graph, excluding the nodes, n, of set S and where the edges are the edges of the starting graph that connect to the S nodes. Colored nodes
form a clique, and the formed clique is then compared with the current maximum clique. If the clique is larger, the new clique is stored as the current maximum clique. The algorithm is shown in Figure 4.

```
procedure BalasYu_MC(s)
    while (SelectSubProblem(s,n))
        SolveSelectedSubProblem(s,n);
        if (SizeSelectedSubProblem(s)>CurrentMCSize) then
            SaveCurrentMC(s);
            CurrentMCSize = SizeSelectedSubProblem(s);
        end if
    end while
    s' = update(s);
    BalasYu_MC(s');
end while
BackTrack(s');
end procedure
```

Figure 4. The Balas-Yu Algorithm [4]

### 2.2. Tool Implementation Using the McGregor Algorithm

The tool presented in this paper is composed of two parts. The first part inputs information about the given protein from different files. The tool constructs the single file in the .fog format. The file contains all the required information for the given protein in the numeric representation. Details about the .fog file used by the tool are mentioned later in this section.

The second part of the tool inputs two protein files which are in .fog format. The tool uses McGregor’s subgraph algorithm to find the maximum common subgraph for two protein graphs. The tool yields the maximum common subgraph of the protein graphs to a file in .fog format.

Residues for the protein sequence are represented as graph vertices with the edges being labeled based on the near-neighbor relationship. The residue’s RASA value signifies the residue being exposed to the surface, or not, based on the threshold value considered. The given RASA value of residues is added as the label of the vertex.
The experimentally obtained protein data consist of information about the given protein stored in three different files. One file “posirecord/negarecord” contains information about the residue index number for the selected amino acids that are significant. The next file is a mapping file with information about different properties associated with the protein. The information in the file includes a symbolic representation of the residue sequence, electrostatic potential, rASA, curvature, residue index, etc. The third file consists of information about the relationship between residues for the given protein. This information is utilized to form edges for a graph that represents the protein. The file-conversion tool extracts information from the three files to represent the required protein in one .fog file. Each .fog file obtained with the file-conversion tool consists of the individual protein’s information about the number of vertices, edges, numeric mapping for residue symbols, rASA, and edge label information. The first line of the .fog file consists of the given protein’s name, followed by the number of vertices and the number of edges. The second line of the file is the numeric representation for each residue in the given protein. Each residue symbol is mapped to the corresponding number with one-to-one mapping in the alphabetical sequence. For example, the residue symbol is simply mapped as A → 1, B → 2 etc. The third line of the .fog file is the rASA value that corresponds to each residue in the second line. The rASA value is simply represented as “1” or “0” based on the rASA value being less than/equal to the threshold or greater than the threshold value of 25, respectively. The rASA value signifies the residue being exposed to the surface or not. The “1” value signifies the residues that are exposed while the “0” signifies non-exposed residues. The last line of the .fog file consists of edge information; each edge is represented as three numbers. The first number in the group represents a source vertex, followed by a destination vertex. The last number signifies
whether the given vertices are sequence neighbors. If the last number is “1”, it signifies sequence neighbors while a “0” signifies non-sequence neighbors.

Figure 5 shows the protein representation in .fog format. The first line of the figure’s .fog file mentions the subgraph’s name. The number of vertices is 3 and signifies that there are three residues in this subgraph. There are four edges to represent the information about contacting residues. The second line mentions 25, 18, and 11 as the numeric representation of residues symbol. The third line lists the rASA values for three residues. The last line represent four edges and the highlighted in group of three numbers represent each edge.

```
#subgraph #3 #4
25 18 11
1 0 0
13 0 1 20 2 10 3 10
```

Figure 5. Protein Structure Representation in .fog Format

The next part of the tool inputs two .fog files that represent proteins and then finds the maximum common subgraph. The maximum common subgraph is also in .fog format. The identified maximum common subgraph can be utilized to predict occurrence in other proteins. The tool is implemented using C++. To find the maximum common subgraph, the McGregor subgraph algorithm is used. A Boost Graph Library (BGL) interface is used to compute the maximum common subgraph and is defined in the header file:
mcgregor_common_subgraphs.hpp.

A BGL is an open library that provides a generic interface to access the graph’s structure but hides the implementation details. The property map interface of the BGL allows the association of various labels as vertex and edge properties on the given graph. Different residue properties and structural relationships among the residues are associated using labels to the graph
vertices and edges. An “Adjacency_list” class in the BGL is used to represent the graph. The “Adjacency_list” class provides the mechanism to control the underlying data structures that are used to represent the graph. The data structure in the “Adjacency_list” consists out-edges of the vertex represented as std::list, vertex as std::vector, type of graph, property map for vertex and property map for edge. With this tool, property maps for the vertices are linked with residue symbols and rASA properties for the residues. The property maps for the edges are linked with the neighboring relationship among residues in the protein’s structure.

An interface to compute the maximum common subgraph requires parameters such as graph1, graph2, the user callback function, “edges_equivalent”, and the “vertices_equivalent” function. Two graphs, graph1 and graph2, are defined as “Adjacency_list” graphs. A user callback function is the function object which is invoked when common subgraphs are found between two graphs, allowing the user to add the required functionalities. The “edges_equivalent” or “vertices_equivalent” is a function predicate that returns the true value whenever equivalent edges or equivalent vertices are identified. These function predicates are defined in the program to compare the desired number of properties that are associated with the graph vertices and edges.
2.3. Overview of the Tool

Table 1. Major classes and methods that are in implemented tool

<table>
<thead>
<tr>
<th>Class/Methods</th>
<th>Description</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>property_map</td>
<td>Real world properties associated with the mathematical representation in the form of a graph. Interface object that specifies each property being accessed using the property-map object.</td>
<td>property_map&lt;Graph,v_property_t&gt;::type new_prop =get(v_property,g)</td>
</tr>
<tr>
<td>adjacency_list</td>
<td>Class that can be used to represent graphs with parameters including the data vertex and edge representation, properties of the vertex, and edge.</td>
<td>adjacency_list&lt;outegelist,vertexlist,graph_type, vertex_properties, egdge_properties, GraphProperties, edgelist&gt;</td>
</tr>
<tr>
<td>mcgregor_common_subgraphs_maximum_unique()</td>
<td>Method to explore the entire search space and call the user_callback function with the discovered largest, unique subgraphs.</td>
<td>void mcgregor_common_subgraphs_maximum_unique( const GraphFirst&amp; graph1, const GraphSecond&amp; graph2, bool only_connected_subgraphs, SubgraphCallback user_callback)</td>
</tr>
<tr>
<td>property_map_equivalent()</td>
<td>Method that compares the vertices and edges between the graphs using property maps.</td>
<td>property_map_equivalent&lt;PropertyMapFirst, PropertyMapSecond&gt;</td>
</tr>
</tbody>
</table>

The tool is implemented using C++ utilizing a BGL. Table 1 lists the classes and methods that are used to implement this tool. The source code is compiled using MinGW distribution including the Boost 1.55.0 library which can be downloaded from:
http://nuwen.net/mingw.html. “mingw-11.6.exe” is a self-extracting archive file that can be downloaded from the given web link. The link clearly describes the procedure for installing required binaries that include BGL files. Once the distribution is installed, the tool’s executable files can be run from the command prompt. The tool is divided into two parts. One executable file, file_conversion, is used to build the .fog file with the required parameter searching from the given protein files. The other file, subgraph_finding, is used to identify the maximum common subgraph between two proteins and to determine if one graph is a subgraph of another one.

An executable file ‘file_conversion’ requires five arguments. The first argument represents the name of protein for which the .fog representation is required. The second argument is the mapping.txt file that consists of protein information, including PDB symbols and rASA. The third argument is the posirecord/negarecod file that contains information about the proteins’ residue indices. The fourth argument is the file that has the contacting information for the protein’s residues. The last argument is the .fog file that is created for the given protein. Figure 6 illustrates file_conversion command with the required arguments’ description.

```plaintext
file_conversion.exe argument1 argument2 argument3 argument4 argument5
file_conversion.exe 1mnnA C:\data\mapping.txt C:\data\posirecord.txt C:\data\contacting_res\1mnn_A
C:\data
```

where ‘1mnnA’ is the name of the protein file,
C:\data\mapping.txt is the mapping file,
C:\data\posirecord.txt is the posirecord file,
C:\data\contacting_res\1mnn_A is the contacting residue file,
C:\data is the location for output file

Figure 6. Command for file_conversion

An executable “subgraph_finding” has three arguments. The first two arguments include two .fog files for which the maximum common subgraph need to be identified. The last argument is the name of the output subgraph file that stores the identified common subgraph.

18
Figure 7 illustrates the `subgraph_finding` command with a description about the required arguments.

```
subgraph_finding argument1 argument2 argument3
subgraph_finding C:\data\graph1.fog C:\data\graph2.fog
C:\data\subgraph_output.fog
```

where, argument1 is the location of first .fog file
argument2 is the location of second .fog file
argument3 is the location of subgraph_output .fog file

Figure 7. Command for subgraph_finding
Figure 8 shows graph representations for sample proteins with values mentioned in the .fog format. Figure 8(a) and 8(b) represent two sample proteins. Figure 8(c) is the result obtained by comparing two sample proteins. The included information is the residue symbol, rASA, and edge label.

Figure 8. (a) Sample Graph 1 with .fog Representation, (b) Sample Graph 2 with .fog Representation, and (c) Output Sample Common Subgraph with .fog Representation
CHAPTER 3. DISCUSSION AND CONCLUSION

The file-conversion tool was used to generate .fog files for protein residues in the “posirecord” file by searching the required property information in the “mapping” file. The contacting residue file for individual proteins was also used to discover the relationship between residues. Extracted protein information was stored in .fog files. The protein’s .fog files were then compared to find the common subgraph using the “subgraph_finding” tool. The tool was successful in identifying the common subgraphs between proteins. Using the “subgraph_finding” tool, a test was performed for different proteins and sample graphs. For the “1nlwA” and “1f4kA” proteins, the tool identified three common residues. Similarly, for “1musA” and “1gdtA”, three common residues were found. The comparison results for proteins are listed in Table 2.

Table 2. The result of a comparison for proteins with the number of common residues found

<table>
<thead>
<tr>
<th>First Protein</th>
<th>Second Protein</th>
<th>Common residues found</th>
</tr>
</thead>
<tbody>
<tr>
<td>1f4kA</td>
<td>1nlwA</td>
<td>3</td>
</tr>
<tr>
<td>1musA</td>
<td>1gdtA</td>
<td>3</td>
</tr>
<tr>
<td>1floA</td>
<td>1ignA</td>
<td>3</td>
</tr>
<tr>
<td>1ulqA</td>
<td>1bdtA</td>
<td>2</td>
</tr>
<tr>
<td>1sxqA</td>
<td>1dh3A</td>
<td>2</td>
</tr>
<tr>
<td>1qrvA</td>
<td>1jb7A</td>
<td>2</td>
</tr>
</tbody>
</table>
Figure 9 shows the console output for the “file_conversion.exe” command which is used to obtain the .fog file for a given protein with the required residue, rASA, and neighboring residue information extracted.

Figure 9. Screenshot for file_conversion Command

Figure 10 shows the console output for the “subgraph_finding.exe” command that illustrates the result of the subgraph found for two proteins: “1floA” and “1ignA”.

Figure 10. Screenshot for subgraph_finding Command
The three dimensional protein structure for two proteins “1floA” and “1ignA”, obtained using Jmol is shown in Figures 11 and 12, respectively. The common residues are highlighted in red, blue, and yellow color.

Figure 11. Protein Structure Diagram for 1floA (Common residues are highlighted in red, blue, and yellow)

Figure 12. Protein Structure Diagram for 1ignA (Common residues are highlighted in red, blue, and yellow)
In Figure 13, common residues that are identified between “1floA” and “1ignA”, are shown using the residues’ Jmol representation. Each residue is represented as the graph vertices, and the relationships between the residues are shown by edges that are represented as lines. The dashed line in the figure indicates three dimensional neighbor residues, and the solid line indicates the sequential neighbors. The figure shows the vertex number along with one of the associated vertex labels.

Figure 13. Common Residues Between Proteins: 1floA and 1ignA (colored in red, blue, and yellow)
This paper gives an overview about the tool to predict the proteins’ functional sites using a graph-based approach. This tool provides the standard representation of a protein in a .fog file with the protein information obtained from different files. Currently, one property (i.e., the rASA value) of the residue is considered for computation. This tool extracts the desired protein information and provides the standard representation of proteins for a graph analysis. Furthermore, the tool provides an easy interface to compare proteins. The results are displayed in the console with the size of subgraph, the vertices, and the edge information. The output in .fog format is stored in the file, making further analysis of a subgraph’s occurrence in the new protein file easier. The tool is useful for study and research. In the future, different protein information such as electrostatic potential, surface curvature, etc., associated with residues can be added as properties for the vertices and edges in the graph representation. These added properties help to improve the prediction’s accuracy. Also, a graphical user interface can be developed for the tool.
BIBLIOGRAPHY


