DEVELOPMENT OF CYTOSCAPE PLUGIN FOR THE AUTOMATION OF SNP ANALYSIS AND THEIR PATHWAYS

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

BioInformatics

under the Supervision of

Dr. TirathaRaj Singh

By

Vikaran Singh Thakur 111513

to

Jaypee University of Information Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Table of Content

List of Figures

Certificate

This is to certify that project report entitled "**Automation Of SNP Analysis And Their Biological Pathways",** submitted by **Vikaran Singh Thakur** in partial fulfillment for the award of degree of Bachelor of Technology in Bioinformatics to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Supervisor's Signature: …………………..

Supervisor's Name: Dr. Tiratha Raj Singh

Designation: Assistant Professor (Senior Grade)

Date: 25.05.2015

Acknowledgement

I, Vikaran Thakur using this opportunity to express my gratitude to everyone who supported me throughout the course of this project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

I would also like to thank my project guide Dr. Tiratha Raj Singh who provided me with the knowledge being required and conductive conditions for my project.

Vikaran Singh Thakur
Date: 25.05.2015

Summary

Motifs are small connected sub networks that a network displays in significantly higher frequencies than would be expected for a random network. They have recently gathered much attention as a concept to uncover structural design principles of complex biological networks. GWAS (genome-wide association study) is another concept getting popular these days. GWA study, or GWAS, also known as whole genome association study (WGA study, or WGAS) or common-variant association study (CVAS), is an examination of many common genetic variants in different individuals to see if any variant is associated with a trait. GWAS typically focus on associations between single-nucleotide polymorphisms (SNPs) and traits like major diseases. To make studies related to SNPs and motifs easier, we have developed a Cytoscape plugin for analyzing network motifs and associated SNPs for the concerned genes in these motifs. Cytoscape is a software for visualization of pathways in a graphical format. Our plugin calculates the significance of all present sub networks and maps SNPs to the genes.

Signature of Student:	Signature of Supervisor:
Name of Student: Vikaran Singh Thakur	Supervisor’s Name: Dr. Tiratha Raj Singh
Date: 25.05.2015	Date: 25.05.2015

1. INTRODUCTION

1.1 Graphs

In computer science, graph is a network of nodes and edges and where two nodes are connected by using an edge, several nodes connected to each other with the help of edges form a graph. In other words a graph is an ordered pair G = (V, E) comprising a set V of vertices or nodes together with a set E of edges.

Types of graphs:

i. Directed Graphs
ii. Undirected Graphs
iii. Cyclic Graphs
iv. Acyclic Graphs

A directed graph consists of edges which have a direction from one node to another node, whereas an undirected graph does not have any direction between nodes. A cyclic graph is formed when a graph consists of one or more cycles within them, an acyclic graph is a graph which has no cycles within.

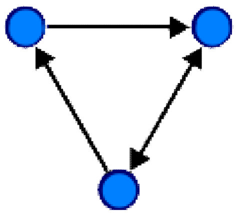


Fig 1.1: Directed Graph

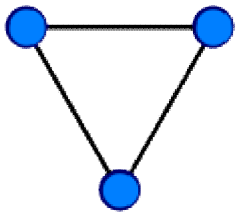


Fig 1.2: Undirected Graph

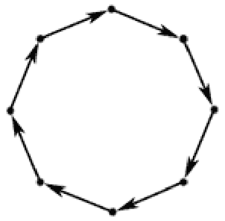


Fig 1.3: Cyclic Graph

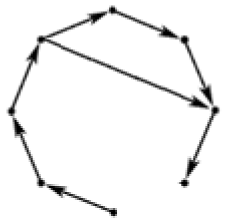


Fig 1.4: Acyclic Graph

1.2 Adjacency Matrix

In mathematics, computer science and application areas such as sociology, an adjacency matrix is a means of representing which vertices (or nodes) of a graph are adjacent to which other vertices. Adjacency matrix is an N x N matrix where N = number of nodes in the graph. It is a way of representing a graph in the form of 0 or 1 (binary numbers). 0 means that there is no edge while 1 means that there is an edge.

Example:

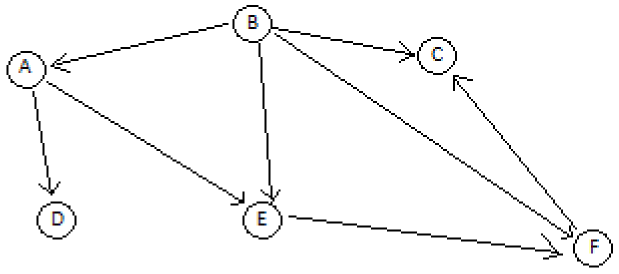


Fig 1.5: An example graph for adjacency matrix

Adjacency matrix for the graph (fig 1.5) would be:

	A	B	C	D	E	F
A	0	0	0	1	1	0
B	1	0	1	0	1	1
C	0	0	0	0	0	0
D	0	0	0	0	0	0
E	0	0	0	0	0	1
F	0	0	1	0	0	0

1.3 Subgraphs

Subgraphs are the graphs within graphs. In other words, a subgraph, H, of a graph, G, is a graph whose vertices are a subset of the vertex set of G, and whose edges are a subset of the edge set of G.

Example:

Adjacency Matrix: 001111100000000000010010001000010000

Vertices: 6

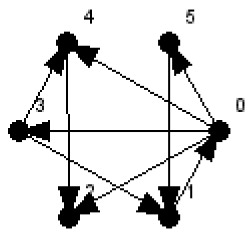


Fig 1.6: Example graph for subgraphs

The above graph will have following 13 subgraphs of node size = 3:

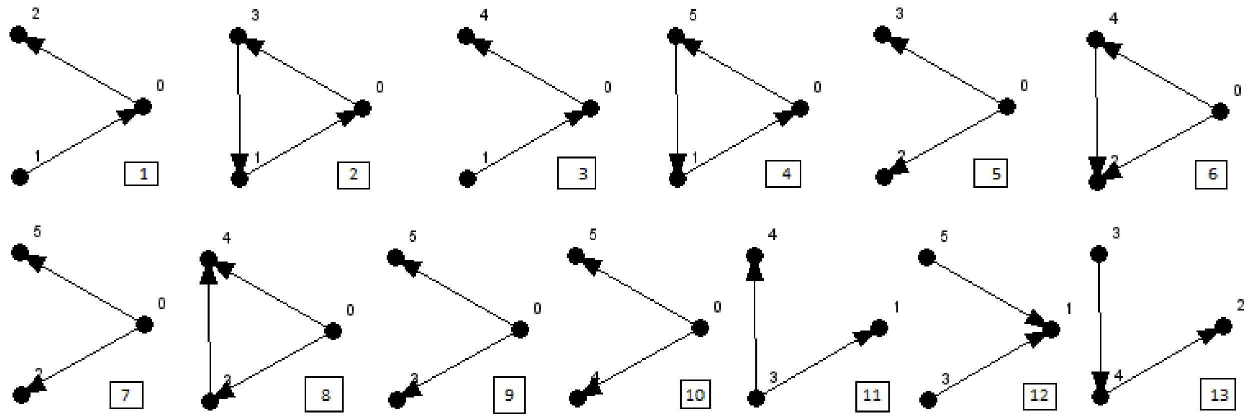


Fig 1.7: Example subgraphs

1.4 Graph Isomorphism

In graph theory, an isomorphism of graphs G and H is a bijection between the vertex sets of G and H

f: V(G) -> V(H)

such that any two vertices u and v of G are adjacent in G if and only if f(u) and f(v) are adjacent in H. This kind of bijection is generally called "edge-preserving bijection", in accordance with the general notion of isomorphism being a structure-preserving bijection.

Therefore to simplify, graph isomorphism occurs when two graphs have:

- the same number of vertices
- the same number of edges
- the same degrees for corresponding vertices
- the same number of connected components
- the same number of loops
- the same number of parallel edges.

In Fig 1.7 graphs numbers 1, 3 and 13 are isomorphic, 2 and 4 are isomorphic, 5,7,9,10 and 11 are isomorphic, and 6 and 8 are isomorphic. Graph number 12 is not isomorphic to any graph.

1.5 Biological Pathways

A biological pathway is a series of actions among molecules in a cell that leads to a certain product or a change in a cell. Such a pathway can trigger the assembly of new molecules, such as a fat or protein. Pathways can also turn genes on and off, or spur a cell to move. Some of the most common biological pathways are involved in metabolism, the regulation of gene expression and the transmission of signals. Pathways play key role in advanced studies of Genomics.

Most common types of biological pathways:

->Metabolic pathway

->Genetic pathway

->Signal transduction pathway

1.6 XML

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. It is defined by the W3C's XML 1.0 Specification and by several other related specifications, all of which are free open standards. The design goals of XML emphasize simplicity, generality and usability across the Internet.

1.7 KEGG and KGML

KEGG (Kyoto Encyclopedia of Genes and Genomes) is a collection of databases dealing with genomes, biological pathways, diseases, drugs, and chemical substances. KEGG is utilized for bioinformatics research and education, including data analysis in genomics, metagenomics, metabolomics and other omics studies, modeling and simulation in systems biology, and translational research in drug development.

KEGG is a database resource for understanding high-level functions and utilities of the biological system, such as the cell, the organism and the ecosystem, from genomic and molecular-level information. It is a computer representation of the biological system, consisting of molecular building blocks of genes and proteins (genomic information) and chemical substances (chemical information) that are integrated with the knowledge on molecular wiring diagrams of interaction, reaction and relation networks (systems information).

The KEGG database has been in development by Kanehisa Laboratories since 1995, and is now a prominent reference knowledge base for integration and interpretation of large-scale molecular data sets generated by genome sequencing and other high-throughput experimental technologies.

KGML is an XML representation of biological pathways in KEGG. The KEGG Markup Language (KGML) is an exchange format of the KEGG pathway maps, which is converted from internally used KGML+ (KGML+SVG) format. KGML enables automatic drawing of KEGG pathways and provides facilities for computational analysis and modeling of gene/protein networks and chemical networks. The KGML files for metabolic pathway maps contain two types of graph object patterns, how boxes (enzymes) are linked by "relations" and how circles (chemical compounds) are linked by "reactions". The KGML files for non-metabolic pathway maps contain only the aspect of how boxes (proteins) are linked by "relations". The prefix for the pathway map identifiers indicates the following:

ko - reference pathway map linked to KO entries (K numbers)
rn - reference pathway map linked to REACTION entries (R numbers)
ec - reference pathway map linked to ENZYME entries (EC numbers)
org (three- or four-letter organism code) - organism-specific pathway map linked to GENES entries (gene IDs)

1.8 Cytoscape

Cytoscape is an open source bioinformatics software platform for visualizing molecular interaction networks and integrating with gene expression profiles and other state data. Additional features are available as plugins. Plugins are available for network and molecular profiling analyses, new layouts, additional file format support and connection with databases and searching in large networks. Plugins may be developed using the Cytoscape open Java software architecture by anyone and plugin community development is encouraged. Cytoscape also has a JavaScript-centric sister project named Cytoscape.js that can be used to analyze and visualize graphs in JavaScript environments, like a browser.

1.9 Network Motifs

All networks, including biological networks, social networks, technological networks (e.g., computer networks and electrical circuits) and more, can be represented as graphs, which include a wide variety of subgraphs. One important local property of networks are so-called network motifs, which are defined as recurrent and statistically significant sub-graphs or patterns.
Network motifs are sub-graphs that repeat themselves in a specific network or even among various networks. Each of these sub-graphs, defined by a particular pattern of interactions between vertices, may reflect a framework in which particular functions are achieved efficiently. Indeed, motifs are of notable importance largely because they may reflect functional properties. They have recently gathered much attention as a useful concept to uncover structural design principles of complex networks. Although network motifs may provide a deep insight into the network's functional abilities, their detection is computationally challenging.

Types of network motifs:

Negative auto-regulation (NAR)

One of simplest and most abundant network motifs in E. coli is negative auto-regulation in which a transcription factor (TF) represses its own transcription. This motif was shown to perform two important functions. The first function is response acceleration. NAR was shown to speed-up the response to signals both theoretically and experimentally. This was first shown in a synthetic transcription network and later on in the natural context in the SOS DNA repair system of E .coli. The second function is increased stability of the auto-regulated gene product concentration against stochastic noise, thus reducing variations in protein levels between different cells.

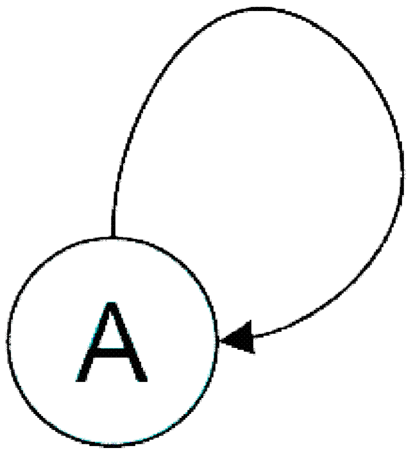


Fig 1.8: Auto-Regulation Motif

Feed-forward loops (FFL)

This motif is commonly found in many gene systems and organisms. The motif consists of three genes and three regulatory interactions. The target gene C is regulated by 2 TFs A and B and in addition TF B is also regulated by TF A. Since each of the regulatory interactions may either be positive or negative there are possibly eight types of FFL motifs. Two of those eight types: the coherent type 1 FFL (C1-FFL) (where all interactions are positive) and the incoherent type 1 FFL (I1-FFL) (A activates C and also activates B which represses C) are found much more frequently in the transcription network of E. coli and yeast than the other six types. In addition to the structure of the circuitry the way in which the signals from A and B are integrated by the C promoter should also be considered. In most of the cases the FFL is either an AND gate (A and B are required for C activation) or OR gate (either A or B are sufficient for C activation) but other input function are also possible.

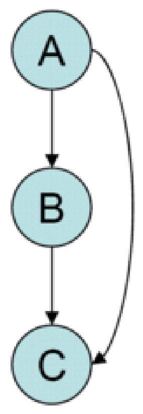


Fig 1.9: Feed-Forward Loop

Coherent type 1 FFL (C1-FFL)

The C1-FFL with an AND gate was shown to have a function of a ‘sign-sensitive delay’ element and a persistence detector both theoretically and experimentally with the arabinose system of E. coli. This means that this motif can provide pulse filtration in which short pulses of signal will not generate a response but persistent signals will generate a response after short delay. The shut off of the output when a persistent pulse is ended will be fast. The opposite behavior emerges in the case of a sum gate with fast response and delayed shut off as was demonstrated in the flagella system of E. coli.

Incoherent type 1 FFL (I1-FFL)

The I1-FFL is a pulse generator and response accelerator. The two signal pathways of the I1-FFL act in opposite directions where one pathway activates Z and the other represses it. When the repression is complete this leads to a pulse-like dynamics. It was also demonstrated experimentally that the I1-FFL can serve as response accelerator in a way which is similar to the NAR motif. The difference is that the I1-FFL can speed-up the response of any gene and not necessarily a transcription factor gene. Recently additional function was assigned to the I1-FFL network motif: it was shown both theoretically and experimentally that the I1-FFL can generate non-monotonic input function in both a synthetic and native systems.

Multi-output FFLs

In some cases the same regulators X and Y regulate several Z genes of the same system. By adjusting the strength of the interactions this motif was shown to determine the temporal order of gene activation. This was demonstrated experimentally in the flagella system of E. coli.

Single-input modules (SIM)

This motif occurs when a single regulator regulates a set of genes with no additional regulation. This is useful when the genes are cooperatively carrying out a specific function and therefore always need to be activated in a synchronized manner. By adjusting the strength of the interactions it can create temporal expression program of the genes it regulates.

In the literature, Multiple-input modules (MIM) arose as a generalization of SIM. However, the precise definitions of SIM and MIM have been a source of inconsistency. There are attempts to provide orthogonal definitions for canonical motifs in biological networks and algorithms to enumerate them, especially SIM, MIM and Bi-Fan (2x2 MIM).

Dense overlapping regulons (DOR)

This motif occurs in the case that several regulators combinatorially control a set of genes with diverse regulatory combinations. This motif was found in E. coli in various systems such as carbon utilization, anaerobic growth, stress response and others. In order to better understand the function of this motif one has to obtain more information about the way the multiple inputs are integrated by the genes. Kaplan et al. has mapped the input functions of the sugar utilization genes in E. coli, showing diverse shapes.

1.10 SNP (Single Nucleotide Polymorphism)

A Single Nucleotide Polymorphism, also known as Simple Nucleotide Polymorphism, (SNP, pronounced snip; plural snips) is a DNA sequence variation occurring commonly within a population (e.g. 1%) in which a single nucleotide — A, T, C or G — in the genome (or other shared sequence) differs between members of a biological species or paired chromosomes. For example, two sequenced DNA fragments from different individuals, AAGCCTA to AAGCTTA, contain a difference in a single nucleotide. In this case we say that there are two alleles. Almost all common SNPs have only two alleles. The genomic distribution of SNPs is not homogenous; SNPs occur in non-coding regions more frequently than in coding regions or, in general, where natural selection is acting and 'fixing' the allele (eliminating other variants) of the SNP that constitutes the most favorable genetic adaptation.

Single-nucleotide polymorphisms may fall within coding sequences of genes, non-coding regions of genes, or in the intergenic regions (regions between genes). SNPs within a coding sequence do not necessarily change the amino acid sequence of the protein that is produced, due to degeneracy of the genetic code.

SNPs in the coding region are of two types, synonymous and nonsynonymous SNPs. Synonymous SNPs do not affect the protein sequence while nonsynonymous SNPs change the amino acid sequence of protein. The nonsynonymous SNPs are of two types: missense and nonsense. A missense mutation (a type of nonsynonymous substitution) is a point mutation in which a single nucleotide change results in a codon that codes for a different amino acid. A nonsense mutation is a point mutation in a sequence of DNA that results in a premature stop codon, or a nonsense codon in the transcribed mRNA, and in a truncated, incomplete, and usually nonfunctional protein product. It differs from a missense mutation, which is a point mutation where a single nucleotide is changed to cause substitution of a different amino acid.

SNPs that are not in protein-coding regions may still affect gene splicing, binding of transcription factor, messenger RNA degradation, or the sequence of non-coding RNA. Gene expression affected by this type of SNP is referred to as an eSNP (expression SNP) and may be upstream or downstream from the gene.

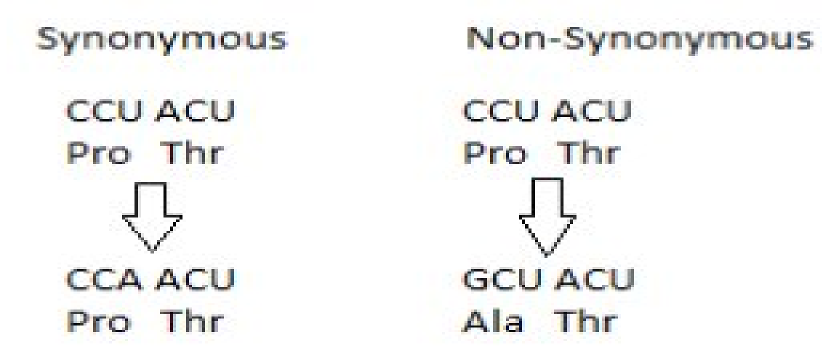


Fig 1.10: Synonymous and Non-Synonymous SNPs

1.11 dbSNP

The Single Nucleotide Polymorphism Database (dbSNP) is a free public archive for genetic variation within and across different species developed and hosted by the National Center for Biotechnology Information (NCBI) in collaboration with the National Human Genome Research Institute (NHGRI). Although the name of the database implies a collection of one class of polymorphisms only (i.e., single nucleotide polymorphisms (SNPs)), it in fact contains a range of molecular variation:

(1) SNPs
(2) short deletion and insertion polymorphisms (indels/DIPs)
(3) microsatellite markers or short tandem repeats (STRs)
(4) multinucleotide polymorphisms (MNPs)
(5) heterozygous sequences
(6) named variants

The dbSNP accepts apparently neutral polymorphisms, polymorphisms corresponding to known phenotypes, and regions of no variation. It was created in September 1998 to supplement GenBank, NCBI's collection of publicly available nucleic acid and protein sequences.

As of build 131 (available February 2010), dbSNP had amassed over 184 million submissions representing more than 64 million distinct variants for 55 organisms, including Homo sapiens, Mus musculus, Oryza sativa, and many other species.

2. AVAILABLE ALGORITHMS TO DETECT NETWORK MOTIFS

2.1 MFINDER

MFINDER, the first motif-mining tool, implements two kinds of motif finding algorithms: a full enumeration and a sampling method. Until 2004, the only exact counting method for NM (network motif) detection was the brute-force one proposed by Milo et al. This algorithm was successful for discovering small motifs, but using this method for finding even size 5 or 6 motifs was not computationally feasible. Hence, a new approach to this problem was needed.

Definitions: Esis the set of picked edges. Vs is the set of all nodes that are touched by the edges in E.

Init Vs and Es to be empty sets.

1. Pick a random edge e1 = (vi, vj). Update Es = {e1}, Vs = {vi, vj}
2. Make a list L of all neighbor edges of Es. Omit from L all edges between members of Vs.
3. Pick a random edge e = {vk,vl} from L. Update Es = Es ∪ {e}, Vs = Vs ∪ {vk, vl}.
4. Repeat steps 2-3 until completing an n-node subgraph (until |Vs| = n).
5. Calculate the probability to sample the picked n-node subgraph.

2.2 FPF (Mavisto)

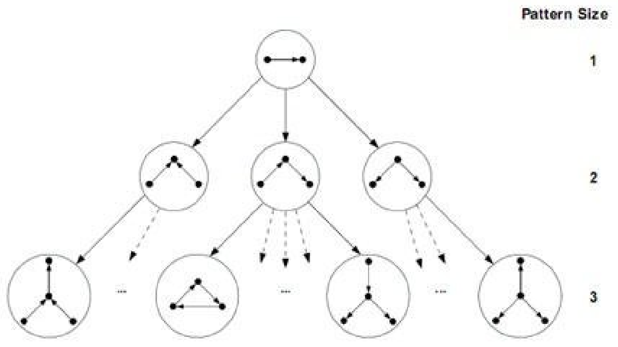


Fig 2.1: Illustration of the pattern tree in FPF algorithm

Schreiber and Schwöbbermeyer proposed an algorithm named flexible pattern finder (FPF) for extracting frequent sub-graphs of an input network and implemented it in a system named Mavisto. Their algorithm exploits the downward closure property which is applicable for frequency concepts F2 and F3. The downward closure property asserts that the frequency for sub-graphs decrease monotonically by increasing the size of sub-graphs; however, this property does not hold necessarily for frequency concept F1. FPF is based on a pattern tree (see Fig 2.1) consisting of nodes that represents different graphs (or patterns), where the parent of each node is a sub-graph of its children nodes; in other words, the corresponding graph of each pattern tree's node is expanded by adding a new edge to the graph of its parent node.

Mavisto

Data: Graph G, target pattern size t, frequency concept F

Result: Set R of patterns of size t with maximum frequency.

```
$R \leftarrow \varphi$, $f_{max} \leftarrow 0$
P ←start pattern p1 of size 1
$M_{p1}$ ←all matches of $p_1$ in G
While $P \neq \varphi$ do
 $P_{max}$ ←select all patterns from P with maximum size.
 P ← select pattern with maximum frequency from $P_{max}$
 E = ExtensionLoop(G, p, $M_p$)
 Foreach pattern $p \in E$
 If $F = F_1$ then $f \leftarrow size(M_p)$
 Else $f \leftarrow$ Maximum Independent set(F, $M_p$)
 End
 If $size(p) = t$ then
 If $f = f_{max}$ then $R \leftarrow R \cup \{p\}$
 Else if $f > f_{max}$ then $R \leftarrow \{p\}$; $f_{max} \leftarrow f$
 End
 Else
```

```
If $F = F_1$ or $f \geq f_{max}$ then $P \leftarrow P \cup \{p\}$
End
End
End
End
```

2.3 ESU (FANMOD)

The sampling bias of Kashtan et al. provided great impetus for designing better algorithms for the NM discovery problem. Although Kashtan et al. tried to settle this drawback by means of a weighting scheme, this method imposed an undesired overhead on the running time as well a more complicated implementation. This tool is one of the most useful ones, as it supports visual options and also is an efficient algorithm with respect to time. But, it has a limitation on motif size as it does not allow searching for motifs of size 9 or higher because of the way the tool is implemented.
Wernicke introduced an algorithm named RAND-ESU that provides a significant improvement over mfinder. This algorithm, which is based on the exact enumeration algorithm ESU, has been implemented as an application called FANMOD. RAND-ESU is a NM discovery algorithm applicable for both directed and undirected networks, effectively exploits an unbiased node sampling throughout the network, and prevents overcounting sub-graphs more than once. Furthermore, RAND-ESU uses a novel analytical approach called DIRECT for determining sub-graph significance instead of using an ensemble of random networks as a Null-model. The DIRECT method estimates the sub-graph concentration without explicitly generating random networks. Empirically, the DIRECT method is more efficient in comparison with the random network ensemble in case of sub-graphs with a very low concentration; however, the classical Null-model is faster than the DIRECT method for highly concentrated sub-graphs.In the following, we detail the ESU algorithm and then we show how this exact algorithm can be modified efficiently to RAND-ESU that estimates sub-graphs concentrations.

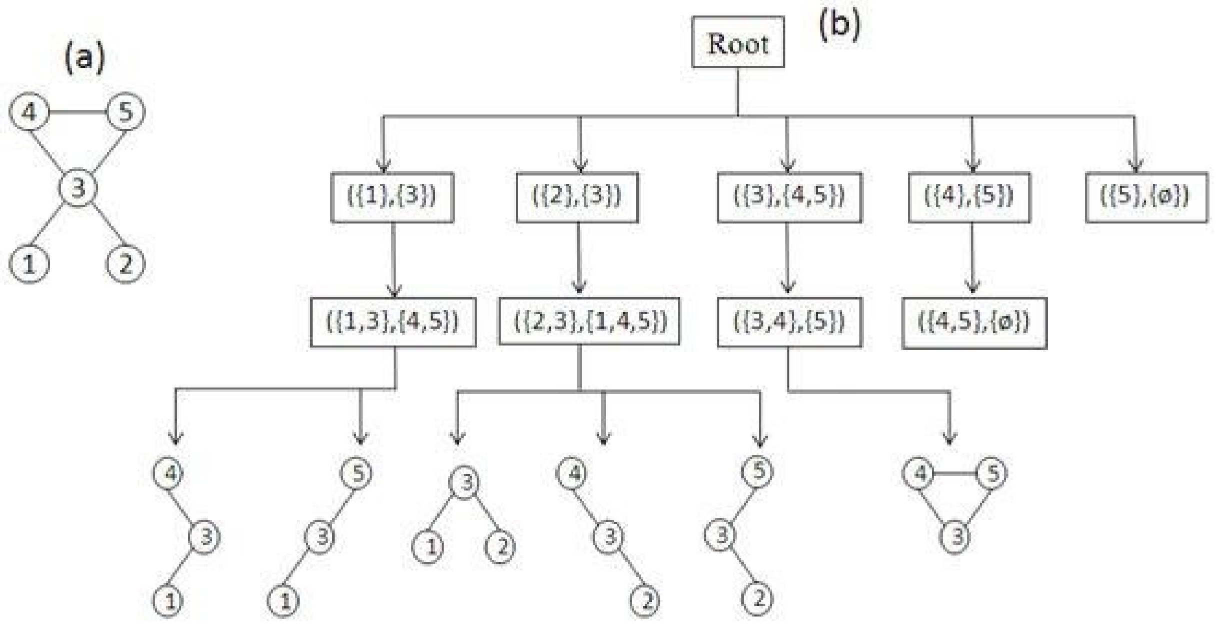


Fig 2.2: RAND-ESU

The algorithms ESU and RAND-ESU are fairly simple, and hence easy to implement. ESU first finds the set of all induced sub-graphs of size *k*, let *Sk* be this set. ESU can be implemented as a recursive function; the running of this function can be displayed as a tree-like structure of depth *k*, called the ESU-Tree (see Fig 2.2).

How has the exact ESU been algorithm modified to RAND-ESU that estimates sub-graph concentrations? The procedure of implementing RAND-ESU is quite straightforward and is one of the main advantages of FANMOD. One can change the ESU algorithm to explore just a portion of the ESU-Tree leaves by applying a probability value 0 ≤ pd ≤ 1 for each level of the ESU-Tree and oblige ESU to traverse each child node of a node in level d-1 with probability pd. This new algorithm is called RAND-ESU. Evidently, when pd = 1 for all levels, RAND-ESU acts like ESU. For pd = 0 the algorithm finds nothing. Note that, this procedure ensures that the chances of visiting each leaf of the ESU-Tree are the same, resulting in unbiased sampling of sub-graphs through the network. The probability of visiting each leaf is ∏dpd and this is identical for all of the ESU-Tree leaves; therefore, this method guarantees unbiased sampling of sub-graphs from the network. Nonetheless, determining the value of pd for *1 ≤ d ≤ k* is another issue that must be determined manually by an expert to get precise results of sub-graph concentrations. While there is no lucid prescript for this matter, the Wernicke provides some general observations that may help in determining p_d values. In summary, RAND-ESU is a very fast algorithm for NM discovery in the case of induced sub-graphs supporting unbiased sampling method. Although, the main ESU algorithm and so the FANMOD tool is known for discovering induced sub-graphs, there is trivial modification to ESU which makes it possible for finding non-induced sub-graphs, too. The pseudo code of ESU (FANMOD) is shown below:

Enumeration of ESU (FANMOD)

```
EnumerateSubgraphs(G,k)
Input: A graph $G = (V, E)$ and an integer $1 \leq k \leq |V|$.
Output: All size-$k$ subgraphs in $G$.
for each vertex $v \in V$ do
    $VExtension \leftarrow \{u \in N(\{v\}) \mid u > v\}$
    call ExtendSubgraph($\{v\}$, VExtension, v)
endfor

ExtendSubgraph(VSubgraph, VExtension, v)
if $|VSubgraph| = k$ then output $G[VSubgraph]$ and return
while $VExtension \neq \emptyset$ do
    Remove an arbitrarily chosen vertex $w$ from VExtension
    $VExtension' \leftarrow VExtension \cup \{u \in N_{excl}(w, VSubgraph) \mid u > v\}$
    call ExtendSubgraph($VSubgraph \cup \{w\}$, VExtension′, v)
return
```

2.4 NeMoFinder

Chen et al. introduced a new NM discovery algorithm called NeMoFinder in 2006, which adapts the idea in SPIN to extract frequent trees and after that expands them into non-isomorphic graphs. NeMoFinder utilizes frequent size-n trees to partition the input network into a collection of size-n graphs, afterward finding frequent size-n sub-graphs by expansion of frequent trees edge-by-edge until getting a complete size-n graph Kn. The algorithm finds NMs in undirected networks and is not limited to extracting only induced sub-graphs. Furthermore, NeMoFinder is an exact enumeration algorithm and is not based on a sampling method. As Chen et al. claim, NeMoFinder is applicable for detecting relatively large NMs, for instance, finding NMs up to size-12 from the whole S. cerevisiae (yeast) PPI network as the authors claimed.

NeMoFinder consists of three main steps. First, finding frequent size-n trees, then utilizing repeated size-n trees to divide the entire network into a collection of size-n graphs, finally, performing sub-graph join operations to find frequent size-n sub-graphs. In the first step, the algorithm detects all non-isomorphic size-n trees and mappings from a tree to the network. In the second step, the ranges of these mappings are employed to partition the network into size-n graphs. Up to this step, there is no distinction between NeMoFinder and an exact enumeration method. However, a large portion of non-isomorphic size-n graphs still remain. NeMoFinder exploits a heuristic to enumerate non-tree size-n graphs by the obtained information from the preceding steps. The main advantage of the algorithm is in the third step, which generates candidate sub-graphs from previously enumerated sub-graphs. This generation of new size-n sub-graphs is done by joining each previous sub-graph with derivative sub-graphs from itself called cousin sub-graphs. These new sub-graphs contain one additional edge in comparison to the previous sub-graphs. However, there exist some problems in generating new sub-graphs: There is no clear method to derive cousins from a graph, joining a sub-graph with its cousins leads to redundancy in generating particular sub-graph more than once, and cousin determination is done by a canonical representation of the adjacency matrix which is not closed under join operation. NeMoFinder is an efficient network motif finding algorithm for motifs up to size 12 only for protein-protein interaction networks, which are presented as undirected graphs. And it is not able to work on directed networks which are so important in the field of complex and biological networks. The pseudocode of NeMoFinder is shown below:

```
Input:
G - PPI network;
N - Number of randomized networks;
K - Maximal network motif size;
F - Frequency threshold;
S - Uniqueness threshold;
Output:
U - Repeated and unique network motif set;
D ← ∅;
for motif-size k from 3 to K do
```

```
    T ← *FindRepeatedTrees*(k);
    GDk ← *GraphPartition*(G, T)
    $D \leftarrow D \cup T$;
    $D' \leftarrow T$;
    $i \leftarrow k$;
    **while** $D'' = \emptyset$ **and** $i \leq k \times (k - 1) / 2$ **do**
        D′ ← *FindRepeatedGraphs*(k, i, D′);
        $D \leftarrow D \cup D'$;
        $i \leftarrow i + 1$;
    **end while**
**end for**
**for** counter i **from** 1 **to** N **do**
    Grand ← *RandomizedNetworkGeneration*();
    **for each** $g \in D$ **do**
        *GetRandFrequency*(g, Grand);
    **end for**
**end for**
$U \leftarrow \emptyset$;
**for each** $g \in D$ **do**
    s ← *GetUniqunessValue*(g);
    **if** $s \geq S$ **then**
        $U \leftarrow U \cup \{g\}$;
    **end if**
**end for**
**return** U
```

2.5 Grochow-Kellis

Grochow and Kellis proposed an exact algorithm for enumerating sub-graph appearances in 2007. The algorithm is based on a motif-centric approach, which means that the frequency of a given sub-graph, called the query graph, is exhaustively determined by searching for all possible mappings from the query graph into the larger network. It is claimed that a motif-centric method in comparison to network-centric methods has some beneficial features. First of all it avoids the increased complexity of sub-graph enumeration. Also, by using mapping instead of enumerating, it enables an improvement in the isomorphism test. To improve the performance of the algorithm, since it is an inefficient exact enumeration algorithm, the authors introduced a fast method which is called symmetry-breaking conditions. During straightforward sub-graph isomorphism tests, a sub-graph may be mapped to the same sub-graph of the query graph multiple times. In the Grochow-Kellis (GK) algorithm symmetry-breaking is used to avoid such multiple mappings.

2.6 Color-Coding Approach

Most algorithms in the field of NM discovery are used to find induced sub-graphs of a network. In 2008, Noga Alon et al. introduced an approach for finding non-induced sub-graphs too. Their technique works on undirected networks such as PPI ones. Also, it counts non-induced trees and bounded treewidth sub-graphs. This method is applied for sub-graphs of size up to 10.

This algorithm counts the number of non-induced occurrences of a tree T with k = O(logn) vertices in a network G with n vertices as follows:

1. Color coding. Color each vertex of input network G independently and uniformly at random with one of the k colors.
2. Counting. Apply a dynamic programming routine to count the number of non-induced occurrences of T in which each vertex has a unique color. For more details on this step.
3. Repeat the above two steps O(ek) times and add up the number of occurrences of T to get an estimate on the number of its occurrences in G.

As available PPI networks are far from complete and error free, this approach is suitable for NM discovery for such networks. As Grochow-Kellis Algorithm and this one are the ones popular for non-induced sub-graphs, it is worth to mention that the algorithm introduced by Alon et al. is less time consuming than the Grochow-Kellis Algorithm.

2.7 MODA

Omidi et al.introduced a new algorithm for motif detection named MODA which is applicable for induced and non-induced NM discovery in undirected networks. It is based on the motif-centric approach discussed in the Grochow-Kellis algorithm section. It is very important to distinguish motif-centric algorithms such as MODA and GK algorithm because of their ability to work as query-finding algorithms. This feature allows such algorithms to be able to find a single motif query or a small number of motif queries (not all possible sub-graphs of a given size) with larger sizes. As the number of possible non-isomorphic sub-graphs increases exponentially with sub-graph size, for large size motifs (even larger than 10), the network-centric algorithms, those looking for all possible sub-graphs, face a problem. Although motif-centric algorithms also have problems in discovering all possible large size sub-graphs, but their ability to find small numbers of them is sometimes a significant property.

```
Input: G: Input graph, k: sub-graph size, Δ: threshold value
Output: Frequent Subgraph List: List of all frequent k-size sub-graphs
Note: $F_G$: set of mappings from G in the input graph G
fetch $T_k$
do
 G′ = Get-Next-BFS($T_k$) // G′ is a query graph
 if |E(G′)| = (k – 1)
 call Mapping-Module(G′, G)
 else
 call Enumerating-Module(G′, G, $T_k$)
 end if
 save $F_2$
 if $|F_G| > Δ$ then
 add G′ into Frequent Subgraph List
 end if
```

```
Until $|E(G')| = (k - 1)/2)$
return Frequent Subgraph List
```

2.8 Kavosh

A recently introduced algorithm named Kavosh aims at improved main memory usage. Kavosh is usable to detect NM in both directed and undirected networks. The main idea of the enumeration is similar to the GK and MODA algorithms, which first find all k-size sub-graphs that a particular node participated in, then remove the node, and subsequently repeat this process for the remaining nodes.

For counting the sub-graphs of size k that include a particular node, trees with maximum depth of k, rooted at this node and based on neighborhood relationship are implicitly built. Children of each node include both incoming and outgoing adjacent nodes. To descend the tree, a child is chosen at each level with the restriction that a particular child can be included only if it has not been included at any upper level. After having descended to the lowest level possible, the tree is again ascended and the process is repeated with the stipulation that nodes visited in earlier paths of a descendent are now considered unvisited nodes. A final restriction in building trees is that all children in a particular tree must have numerical labels larger than the label of the root of the tree. The restrictions on the labels of the children are similar to the conditions which GK and ESU algorithm use to avoid overcounting sub-graphs.

The protocol for extracting sub-graphs makes use of the compositions of an integer. For the extraction of sub-graphs of size k, all possible compositions of the integer k-1 must be considered. The compositions of k-1 consist of all possible manners of expressing k-1 as a sum of positive integers. Summations in which the order of the summands differs are considered distinct. A composition can be expressed as k2,k3,…,km where k2 + k3 + … + km = k-1. To count sub-graphs based on the composition, ki nodes are selected from the i-th level of the tree to be nodes of the sub-graphs (i = 2,3,…,m). The k-1 selected nodes along with the node at the root define a sub-graph within the network. After discovering a sub-graph involved as a match in the target network, in order to be able to evaluate the size of each class according to the target network, Kavosh employs the nauty algorithm in the same way as FANMOD. The enumeration part of Kavosh algorithm is shown below:

```
***Enumerate_Vertex(G, u, S, Remainder, i)***

**Input:** G: Input graph
 u: Root vertex
 S: selection (S = { $S_0$,$S_1$,...,$S_{k-1}$} is an array of the set of all $S_i$)
 Remainder: number of remaining vertices to be selected
 i: Current depth of the tree.
**Output:** all k-size sub-graphs of graph G rooted in u.

**if** Remainder = 0 **then**
 **return**
**else**
 ValList ← *Validate*(G, $S_{i-1}$, u)
 $n_i$ ← *Min*(|ValList|, Remainder)
 **for** $k_i$ = 1 **to** $n_i$ **do**
 C ← *Initial_Comb*(ValList, $k_i$)
 (Make the first vertex combination selection according)
 **repeat**
 $S_i$ ← C
 *Enumerate_Vertex*(G, u, S, Remainder-$k_i$, i+1)
 *Next_Comb*(ValList, $k_i$)
 (Make the next vertex combination selection according)
 **until** C = NILL
 **end for**
 **for each** v ∈ ValList **do**
 Visited[v] ← false
 **end for**
**end if**
```

```
***Validate(G, Parents, u)***

**Input:** G: input graph, Parents: selected vertices of last layer, u: Root vertex.
**Output:** Valid vertices of the current level.

ValList ← NILL
**for each** v ∈ Parents **do**
 **for each** w ∈ Neighbor[u] **do**
 **if** label[u] < label[w] **AND NOT** Visited[w] **then**
 Visited[w] ← true
 ValList = ValList + w
 end if
 end for
end for
return ValList
```

2.9 G-Tries

In 2010, Pedro Ribeiro and Fernando Silva proposed a novel data structure for storing a collection of sub-graphs, called a G-Tries. This data structure, which is conceptually akin to a prefix tree, stores sub-graphs according to their structures and finds occurrences of each of these sub-graphs in a larger graph. One of the noticeable aspects of this data structure is that coming to the network motif discovery, the sub-graphs in the main network are needed to be evaluated. So, there is no need to find the ones in random network which are not in the main network. This can be one of the time-consuming parts in the algorithms in which all sub-graphs in random networks are derived.
G-Tries is a multiway tree that can store a collection of graphs. Each tree node contains information about a single graph vertex and its corresponding edges to ancestor nodes. A path from the root to a leaf corresponds to one single graph. Descendants of G-Tries node share a common sub-graph. Constructing G-Tries is well described in. After constructing a G-Tries, the counting part takes place. The main idea in counting process is to backtrack by all possible sub-graphs, but at the same time do the isomorphism tests. This backtracking technique is essentially the same technique employed by other motif-centric approaches like MODA and GK algorithms. Taking advantage of common substructures in the sense that at a given time there is a partial isomorphic match for several different candidate sub-graphs.

2.10 Comparison

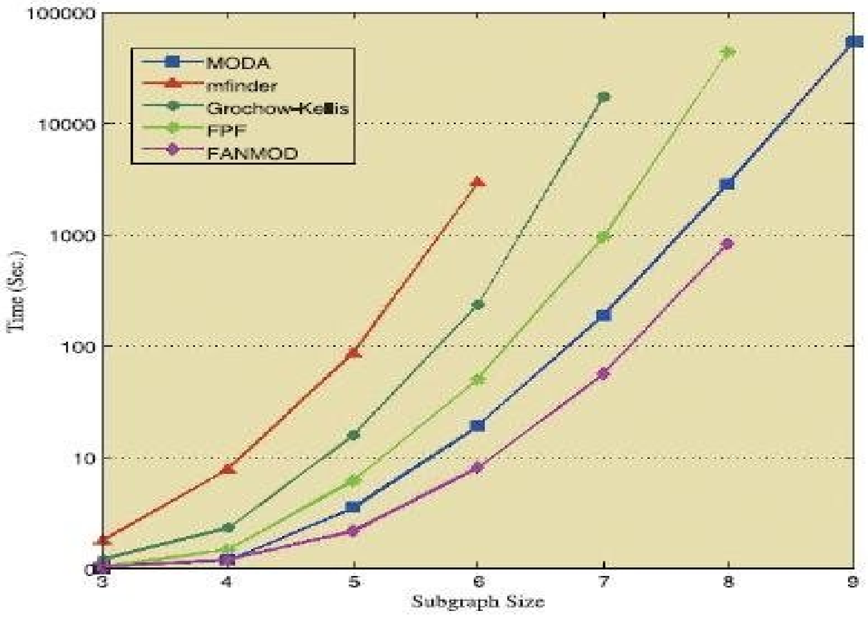


Fig 2.3: Comparison of algorithms.

By looking at Fig 2.3, it is clear that FANMOD is the fastest algorithm for computing network motifs.

Comparison of FANMOD, Kavosh, Mavisto and MFinder for E.Coli (Time in seconds)

NodeSize > Algorithms: \/	3	4	5	6	7	8	9	10
Kavosh	0.30	1.84	14.91	141.98	1374.0	13173.7	121110.3	1120560.1
FANMOD	0.81	2.53	15.71	132.24	1205.9	9256.6	-	-
Mavisto	13532	-	-	-	-	-	-	-
MFinder	31.0	297	23671	-	-	-	-	-

3. METHODOLOGIES

3.1 Overview

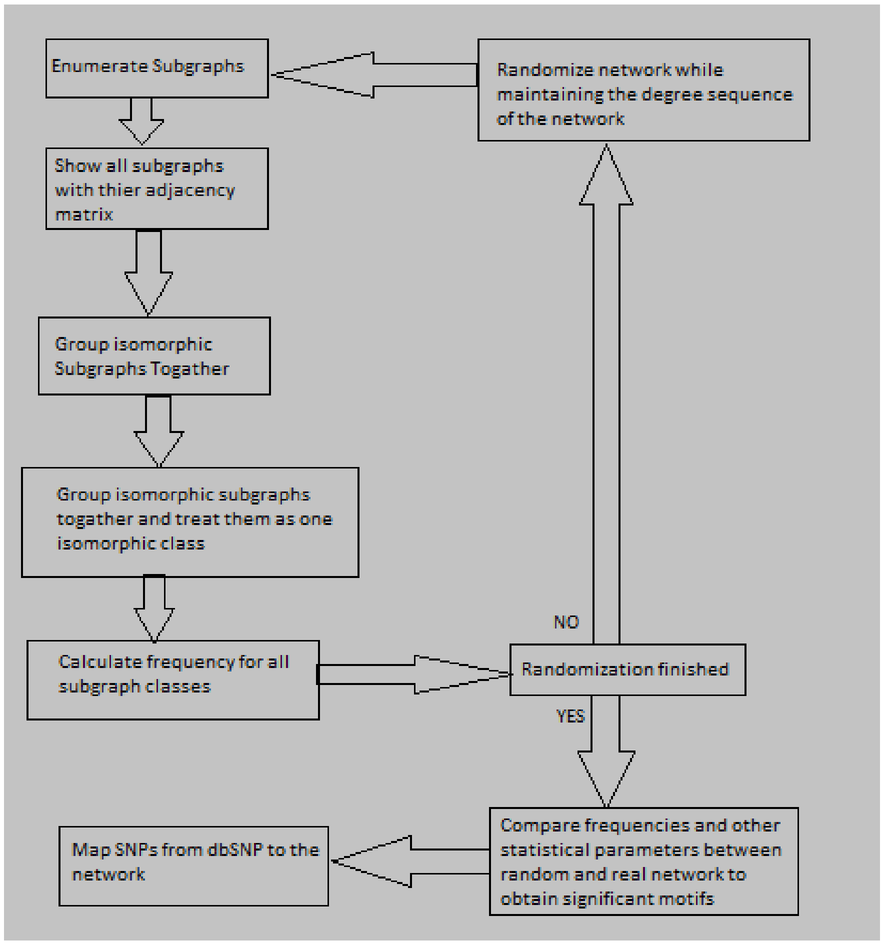
Detection of network motifs can be classified to following three subtasks:

(i) Enumeration of subgraphs.

(ii) Detection of isomorphic subgraphs.

(iii) Randomization of networks.

Our approach is similar to FANMOD, however there is some modification in the first and second subtasks. Once the network motifs are detected, SNPs are mapped on to the network using dbSNP, SNP data is automatically obtained through the internet from dbSNP.



Enumerate Subgraphs
Randomize network while maintaining the degree sequence of the network
Show all subgraphs with thier adjacency matrix
Group isomorphic Subgraphs Togather
Group isomorphic subgraphs togather and treat them as one isomorphic class
NO
Calculate frequency for all subgraph classes
Randomization finished
YES
Map SNPs from dbSNP to the network
Compare frequencies and other statistical parameters between random and real network to obtain significant motifs

Fig 3.1: Overview of Methodology

3.2 Subgraph Enumeration

An approach to traverse through all possible subgraphs is used based on permutations and combinations.

Algorithm to get subgraphs:

S: Set of nodes in the subgraph

K: Size of subgraph

N: Number of nodes in the network

findSubgraphs (S, K)

Input: An empty linked list S and subgraph size K.

Output: All size-k subgraphs in G.

```
findSubgraphs (S, K) {
        if(k==0) {
                if(new subgraph) {
                        addsubgraph(S);
                        Empty the set S
                }
        }
        else {
                for all N nodes {
                        if(unvisited node) {
                                if(node is connected) {
                                        add node to S
                                        findSubgraphs (S, K-1)
                                }
                        }
                }
        }
}
```

This is an exact non-polynomial algorithm and will find all possible size-K subgraphs. It runs with a time complexity of O(n^k).

3.3 Detection of Subgraph Isomorphism

Over the last 30 years, researchers have been trying to find out the best solution to detect subgraph isomorphism. The best known algorithm till date is nauty algorithm, however here we developed an algorithm of our own to detect graph isomorphism in the best possible way.

Steps involved to detect subgraph isomorphism:

1. Compute subgraph adjacency matrix.
2. Get decimal value for adjacency matrix (which is in binary format)
3. Rotate the subgraph to obtain a new adjacency matrix and get the decimal value for that.
4. Compare the current decimal value of the matrix with the previous value of the matrix.
5. If the value is less than previous then store the current value.
6. Repeat step 3 until all possible rotations are done for the subgraph.

This will give us a unique adjacency matrix for each subgraph class.

Proof:

Consider two isomorphic graphs A and B.

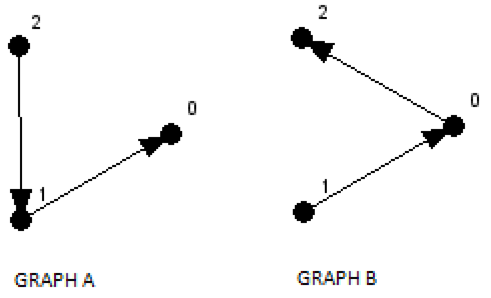


Fig 3.2: Example Isomorphic Graphs

Adjacency Matrix for A: 001100000

Adjacency Matrix for B: 000100010

On rotating the graphs, we get following matrices,

For A: 010000100,010001000,001000010,000100010,000001100,001100000

For B: 000001100,010000100,001100000,001000010,010001000,000100010

We can already note that few adjacency matrices are common for both the graphs, which proves they are isomorphic. But to detect isomorphism we will have to compare the matrices with each other which is computationally expensive. We need an approach where we do not need to compare graphs with each other, an approach which can give us a unique id to represent all graphs which are isomorphic to each other.

On converting the above matrices to decimal values for both A and B we get:

For A: 96,132,136,66,34,12,….

For B: 34,12,132,96,66,136,…

Therefore the smallest value for adjacency matrix is 12, this is the unique id for the particular isomorphic class and graphs can be grouped together using this id. Another approach would be to take the highest value, i.e. 136, but higher values need more storage, hence we go for the smallest values.

Algorithm:

getIsomorphicClass (S,k,classid)

Input: An empty linked list, subgraph size and decimal representation of adjacency matrix.

Output: Unique class id for the isomorphic class.

```
getIsomorphicClass (S,k,classid) {
    if($k==0) {
        dec=binarytodecimal(adjacencymatrix);
        if(dec<classid)
            classid=dec;
        return $classid;
    }
    else {
        for all nodes
            if(node is unvisited)
                classid=getIsomorphicClass (S,k-1,classid);
        return classid;
    }
}
```

3.4 Randomization of Network

This is similar to FANMOD. Graphs are randomized while maintaining their degree sequence.
In graph theory, the degree (or valency) of a vertex of a graph is the number of edges incident to the vertex, with loops counted twice. The degree of a vertex v is denoted deg(v). The maximum degree of a graph G, denoted by Δ(G), and the minimum degree of a graph, denoted by δ(G), are the maximum and minimum degree of its vertices. In the graph on the right, the maximum degree is 5 and the minimum degree is 0. In a regular graph, all degrees are the same, and so we can speak of the degree of the graph.

3.5 Statistical Parameters for Detecting Motif Significance

To calculate motif significance following parameters are calculated:

Mean Frequency for Random Networks: The mean frequency with which the motif occurred in random networks.
Standard Deviation: Standard deviation of random networks frequency from the mean frequency of random networks.
Z-Score: The Z-Score is the original frequency minus the random frequency divided by the standard deviation.

3.6 Mapping of SNPs

Finally after detecting motifs for the biological pathway network, we need to map SNPs on the nodes (which are genes). To make this successful, we have developed an API which will give us SNPs related to a particular genes. It can be accessed using the following URL: http://vikaran.in/snp/gene=[nameofgene]
The SNPs are directly obtained from dbSNP at NCBI.

4. CYTOSCAPE PLUGIN: SNP MAPPER

4.1 Overview

SNPMapper is a plugin for Cytoscape and successfully implements all the above discussed methods.

SNPMapper detects network motifs as well as maps SNPs on to the biological pathways. KGML formats can be read in Cytoscape as well using the KEGGScape plugin.

4.2 Advantages of a Cytoscape Plugin

Cytoscape is the best software for visualizing of networks and graphs, i.e. biological pathways. User can directly add/remove/edit nodes (i.e. genes) using a mouse. Each node consists of node attributes and node attributes can be added or removed easily. Cytoscape provides a user friendly environment and has a wide variety of plugins in Cytoscape App Store to enhance Cytoscape functionalities and user experience.

4.3 Installation

For SNPMapper to work, the user needs to have at least Cytoscape v3.0. It will not work for older versions.

SNPMapper consists of two files:

1. SNPMapper.jar
2. SNPMapperConfiguration.ini

The ini file must be placed in the Cytoscape directory.

The jar file can be installed from the App Manager in the Cytoscape.

To get the most out of SNPMapper, you should also install KEGGScape which is a plugin for reading KGML files so that you can directly analyze KGML files using SNPMapper.

4.4 Usage

To use SNPMapper, first you need to set your preferred configuration from the SNPMapperConfiguration.ini.

Configuration:

The configuration file consists of following properties:

SubgraphSize: The total number of nodes in the network motif

EdgesToRandomize: To replace the number of edges to generate random graphs.

TotalNumberOfRandomNetworks: The total number of random networks to be generated.

Importing KGML And Mapping SNPs:

If you are using KEGGScape plugin, you can directly import KGML files from the File menu in Cytoscape. KGML files can be downloaded from the KEGG Pathways Database. After that click on "Apps" menu and click on "Detect Motifs and Map SNPs".

This will generate network motifs and map SNPs on the biological pathway.

4.5 Results

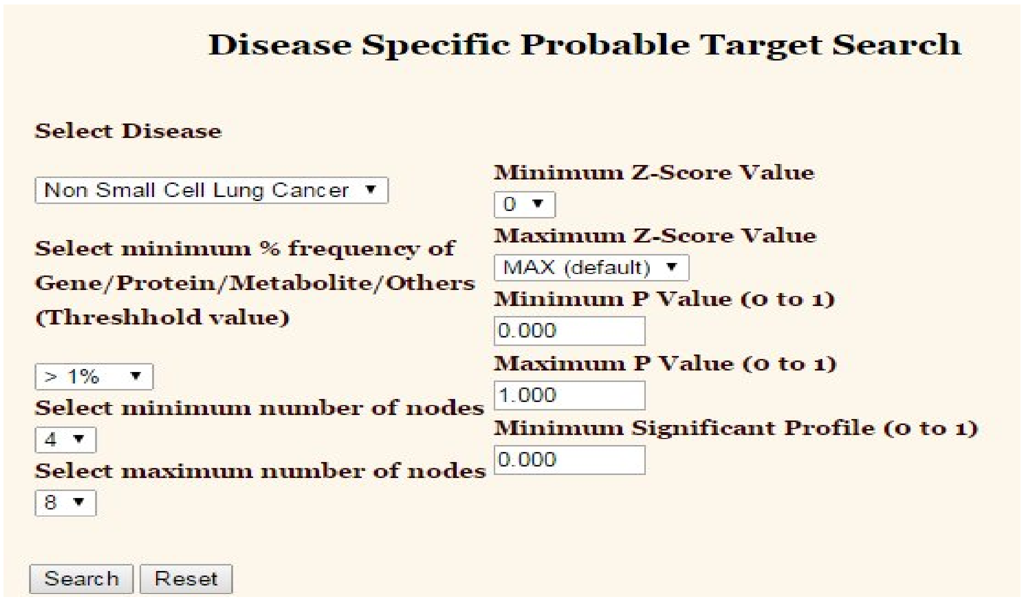
The plugin generates an HTML dump file where all the motifs can be viewed along with statistical parameters. The SNPs are directly mapped on the network in Cytoscape. The node attribute table can be imported as a CSV file in Cytoscape from the file menu, export option.

5. WORK DONE IN FANMOD

5.1 Overview

5 diseases (mentioned below) were analyzed in FANMOD and their network motifs were obtained. Network Motifs were generated using the FANMOD, number of random network motifs were kept to 1000, node size: 3 to 8. Frequency for all genes calculated for a particular disease. Total number of all the genes involved in a particular motif was calculated. Total number of different genes involved in a particular motif was calculated. Keeping a certain threshold value for the diseases, probable targets were predicted.

A web interface was made where user could visualize network motifs for all those 5 diseases in an organized manner, basically user can view FANMOD results for all 5 diseases in an organized manner along with 3 other parameters as mentioned above.



Disease Specific Probable Target Search

Select Disease: Non Small Cell Lung Cancer

Select minimum % frequency of Gene/Protein/Metabolite/Others (Threshhold value): > 1%

Select minimum number of nodes: 4

Select maximum number of nodes: 8

Minimum Z-Score Value: 0

Maximum Z-Score Value: MAX (default)

Minimum P Value (0 to 1): 0.000

Maximum P Value (0 to 1): 1.000

Minimum Significant Profile (0 to 1): 0.000

Fig 5.1: Disease Specific Probable Target Search

5.2 Diseases

1. Non-Small Cell Lung Cancer
2. Small Cell Lung Cancer
3. Pancreatic Cancer
4. Prostate Cancer
5. Renal Cancer

5.3 Results

Name of Cancer	Threshold value of frequency of genes in the diseases	Probable Targets
Non-Small Cell Lung Cancer	5	PIK3CA,AKT3,ERBB2,EGFR,GRB2,EGF, TGFA,FOXO3,BAD,CASP9
Small Cell Lung Cancer	7	MAPK1, CDKN2A, CCND1 ,AKT3 , RASSF5,PIK3CA,GRB2,RXRA,RASSF1
Pancreatic Cancer	7	PRKCA, C00076*, C01245*, PDPK1, E2F1, C00165*
Prostate Cancer	7	MAP2K1, CASP9, C05981*, CCND1, BAD, KRAS, RXRA, STK4, RASSF1
Renal Cancer	7	CCND1, C05981*, RASSF1, RASSF5, FHIT, FOXO3, EGF

6. CONCLUSION AND FUTURE ASPECTS

i. Here we initially used 5 diseases which are Non-Small Cell Lung Cancer, Small Cell Lung Cancer, Pancreatic Cancer, Prostate Cancer and Renal Cancer, this kind of approach can be implemented on other diseases too.

ii. It will help in reducing the work of researchers working in cancer drugs as the most probable targets can be predicted and based upon these targets putative drugs could be designed.

iii. It will help to study the SNPs relation with myriad of diseases through establishsing a correlation between concerned genes and their SNPs.

iv. Easy to use tool will help biologists and medical scientists to work on diseases, SNPs, and pathways altogether.

7. REFERENCES

1. Milo R, Shen-Orr SS, Itzkovitz S, Kashtan N, Chklovskii D, Alon U (2002). "Network motifs: simple building blocks of complex networks". Science 298 (5594): 824–827. doi:10.1126/science.298.5594.824. PMID 12399590
2. Rowan Christmas, Iliana Avila-Campillo, Hamid Bolouri, Benno Schwikowski, Mark Anderson, Ryan Kelley, Nerius Landys, Chris Workman, Trey Ideker, Ethan Cerami, Rob Sheridan, Gary D. Bader, and Chris Sander "Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks" Am Assoc Cancer Res Educ Book 2005: 12-16
3. Kanehisa, Minoru; Goto, Susumu; Sato, Yoko; Furumichi, Miho; Tanabe, Mao "KEGG - Kyoto Encyclopedia of Genes and Genomes" http://www.oxfordjournals.org/our_journals/nar/database/summary/112
4. Kitts A, and Sherry S, (2009). "The single nucleotide polymorphism database (dbSNP) of nucleotide sequence variation" http://www.ncbi.nlm.nih.gov/books/NBK21088/
5. Kashtan N, Itzkovitz S, Milo R, Alon U (2004). "Efficient sampling algorithm for estimating sub-graph concentrations and detecting network motifs". Bioinformatics 20 (11): 1746–1758. doi:10.1093/bioinformatics/bth163
6. Shen-Orr SS, Milo R, Mangan S, Alon U (May 2002). "Network motifs in the transcriptional regulation network of Escherichia coli". Nat. Genet. 31 (1): 64–8. doi:10.1038/ng881. PMID 11967538
7. Milo R, Shen-Orr S, Itzkovitz S, Kashtan N, Chklovskii D, Alon U (October 2002). "Network motifs: simple building blocks of complex networks". Science 298 (5594): 824–7. doi:10.1126/science.298.5594.824. PMID 12399590
8. Schreiber F, Schwobbermeyer H (2005). "MAVisto: a tool for the exploration of network motifs". Bioinformatics 21 (17): 3572–3574. doi:10.1093/bioinformatics/bti556
9. Chen J, Hsu W, Li Lee M et al. (2006). NeMoFinder: dissecting genome-wide protein-protein interactions with meso-scale network motifs. the 12th ACM SIGKDD international

conference on Knowledge discovery and data mining. Philadelphia, Pennsylvania, USA. pp. 106–115

10. Huan J, Wang W, Prins J et al. (2004). SPIN: mining maximal frequent sub-graphs from graph databases. the 10th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 581–586
11. Grochow JA (2006). On the structure and evolution of protein interaction networks (PDF). Thesis M. Eng., Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science.
12. Kashani ZR, Ahrabian H, Elahi E, Nowzari-Dalini A, Ansari ES, Asadi S, Mohammadi S, Schreiber F, Masoudi-Nejad A (2009). "Kavosh: a new algorithm for finding network motifs". BMC Bioinformatics 10 (318). doi:10.1186/1471-2105-10-318
13. Ali Masoudi-Nejad, Mitra Anasariola, Ali Salehzadeh-Yazdi, Sahand Khakabimamaghani (2012). "CytoKavosh: a Cytoscape Plug-in for Finding Network Motifs in Large Biological Networks". PLOS ONE, in press.
14. Ribeiro P, Silva F (2010). G-Tries: an efficient data structure for discovering network motifs. ACM 25th Symposium On Applied Computing - Bioinformatics Track. Sierre, Switzerland. pp. 1559–1566.
15. Sebastian Wernicke and Florian Rasche "FANMOD: A tool for fast motif network detection". http://bioinformatics.oxfordjournals.org/content/22/9/1152.full
16. Sebastian Wenicke "A Faster Algorithm for Detecting Network Motifs". http://theinf1.informatik.uni-jena.de/publications/network-motifs-wabi05.pdf
17. Zhang K, Chang S, Cui S, Guo L, Zhang L, Wang J "ICSNPathway: identify candidate causal SNPs and pathways from genome-wide association study by one analytical framework." http://www.ncbi.nlm.nih.gov/pubmed/21622953
18. McKay BD (1981). "Practical graph isomorphism". Congressus Numerantium 30: 45–87
19. McKay BD (1998). "Isomorph-free exhaustive generation". Journal of Algorithms 26: 306–324. doi:10.1006/jagm.1997.0898