www.ijiarec.com

ISSN:2348-2079

Volume-7 Issue-1

International Journal of Intellectual Advancements and Research in Engineering Computations

# Effective chronic disease progression model

**Mrs. K.E. Eswari M.C.A., M.Phil.,M.E[1], Mr. R.Kavinkumar[2]**

[1]Assistant Professor, [2]Final MCA Students

Department of MCA, Nandha Engineering College (Autonomous), Erode-52.

TamilNadu. India.

## ABSTRACT

Healthcare insurance fraud has caused billions of dollars in losses in public healthcare funds around the world. In particular, healthcare insurance fraud in chronic diseases is especially rampant. Understanding disease progression can help investigators detect healthcare insurance frauds early on. Existing disease progression methods often ignore complex relations, such as the time-gap and pattern of disease occurrence. They also do not take into account the different medication stages of the same chronic disease, which is of great help when conducting healthcare insurance fraud detection and reducing healthcare costs. This project proposes a heterogeneous network-based chronic disease progression mining method to improve the current understanding on the progression of chronic diseases, including orphan diseases. The method also considers the different medication stages of the same chronic disease. The experiments show that the new method can outperform the existing methods.

**Keywords:** Data Mining, Map Reduces Model, Classification Model, Clustering, Automatic Analysis

## INTRODUCTION

Information mining, or learning revelation, is the PC helped procedure of burrowing through and breaking down tremendous arrangements of information and afterward removing the importance of the information. Information mining apparatuses anticipate practices and future patterns, enabling organizations to make proactive, learning driven choices. Information mining apparatuses can respond to business addresses that customarily were too tedious to determine. They scour databases for shrouded examples, finding prescient data that specialists may miss since it lies outside their desires. Information mining gets its name from the likenesses between looking for important data in a vast database and digging a mountain for a vein of significant mineral. The two procedures require either filtering through a tremendous measure of material, or wisely testing it to discover where the esteem dwells. Despite the fact that information mining is still in its earliest stages, organizations in a wide scope of businesses - including retail, fund, human services, producing transportation, and aviation - are as of now utilizing information mining instruments and procedures to exploit recorded information. By utilizing design acknowledgment innovations and measurable and scientific methods to filter through warehoused data, information mining enables examiners to perceive noteworthy actualities, connections, patterns, examples, special cases and peculiarities that may somehow or another go unnoticed. For organizations, information mining is utilized to find examples and connections in the information so as to help settle on better business choices. Information mining can help spot deals patterns, create more astute promoting efforts, and precisely anticipate client dedication [1].

### Specific uses of data mining include

- Market segmentation - Identify the common characteristics of customers who buy the same products from the company.
- Customer churn - Predict which customers are likely to leave the company and go to a competitor.

**Author for correspondence:**

Department of MCA, Nandha Engineering College (Autonomous), Erode-52. TamilNadu. India.

- Fraud detection - Identify which transactions are most likely to be fraudulent.
- Direct marketing - Identify which prospects should be included in a mailing list to obtain the highest response rate.
- Interactive marketing - Predict what each individual accessing a Web site is most likely interested in seeing.
- Market basket analysis - Understand what products or services are commonly purchased together; e.g., beer and diapers.
- Trend analysis - Reveal the difference between a typical customer this month and last [2-5].

## Automated Prediction of Trends And Behaviors

Data mining automates the process of finding predictive information in a large database. Questions that traditionally required extensive hands-on analysis can now be directly answered from the data. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events [6-8].

## Automated discovery of previously unknown patterns:

Data mining tools sweep through databases and identify previously hidden patterns. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors. While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks. Generally, any of four types of relationships are sought [9].

- ✓ Classes: Stored data is used to locate data in predetermined groups. For example, a restaurant chain could mine customer purchase data to determine when customers visit and what they typically order. This information could be used to increase traffic by having daily specials.
- ✓ Clusters: Data items are grouped according to logical relationships or consumer preferences. For example, data can be mined to identify market segments or consumer affinities.
- ✓ Associations: Data can be mined to identify associations. The beer-diaper example is an example of associative mining.
- ✓ Sequential patterns: Data is mined to anticipate behavior patterns and trends. For example, an outdoor equipment retailer could predict the likelihood of a backpack being purchased based on a consumer's purchase of sleeping bags and hiking shoes.

## Data mining consists of five major elements

- ❖ Extract, transform, and load transaction data onto the data warehouse system.
- ❖ Store and manage the data in a multidimensional database system.
- ❖ Provide data access to business analysts and information technology professionals.
- ❖ Analyze the data by application software.
- ❖ Present the data in a useful format, such as a graph or table [10].

## Different levels of analysis are available

- ✓ Artificial neural networks: Non-linear predictive models that learn through training and resemble biological neural networks in structure.
- ✓ Genetic algorithms: Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of natural evolution.
- ✓ Decision trees: Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic

Interaction Detection (CHAID) . CART and CHAID are decision tree techniques used for classification of a dataset. They provide a set of rules that you can apply to a new (unclassified) dataset to predict which records will have a given outcome. CART segments a dataset by creating 2-way splits while CHAID segments using chi square tests to create multi-way splits. CART typically requires less data preparation than CHAID.

- ✓ Nearest neighbor method: A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset (where k 1). Sometimes called the k-nearest neighbor technique.
- ✓ Rule induction: The extraction of useful if-then rules from data based on statistical significance.
- ✓ Data visualization: The visual interpretation of complex relationships in multidimensional data. Graphics tools are used to illustrate data relations.

## RELATED WORKS

Jeffrey Dean, Sanjay Ghemawat portrays Map decrease is a programming model and a related execution for handling and producing expansive informational collections. Clients indicate a guide work that forms a key/esteem pair to create a lot of moderate key/esteem sets, and a decrease work that combines every single middle esteem related with a similar transitional key. Numerous certifiable assignments are expressible in this model, as appeared in the paper. Projects written in this utilitarian style are consequently parallelized and executed on an expansive group of product machines. The run-time framework deals with the subtleties of apportioning the information, planning the program's execution over a lot of machines, taking care of machine disappointments, and dealing with the required between machine correspondence. This permits software engineers with no involvement with parallel and disseminated frameworks to effectively use the assets of a vast appropriated framework. Our execution of Map Reduce keeps running on an expansive group of product machines and is exceedingly versatile: a commonplace Map Reduce calculation forms numerous terabytes of information on a great many machines. Software engineers discover the framework simple to utilize: many Map Reduce programs have been actualized and upwards of one thousand Map Reduce employments are executed on Google's bunches each day.As a response to this multifaceted nature, we structured another deliberation that enables us to express the basic calculations we were attempting to perform yet shrouds the chaotic subtleties of parallelization, adaptation to non-critical failure, information conveyance and burden adjusting in a library. Our reflection is motivated by the guide and diminishes natives displaying Lisp and numerous other practical dialects. We understood that the majority of our calculations included applying a guide task to each legitimate "record" in our contribution to request to process a lot of moderate key/esteem sets, and afterward applying a decrease activity to every one of the qualities that common a similar key, so as to join the determined information suitably. Our utilization of an utilitarian model with client indicated guide and decrease tasks enables us to parallelize vast calculations effectively and to utilize re-execution as the essential system for adaptation to non-critical failure.

Jie Tang, Jimeng Sun,Chi Wang and Zi Yang. In large social networks, nodes (users, entities) are influenced by others for various reasons. For example, the colleagues have strong influence on one’s work, while the friends have strong influence on one’s daily life. How to differentiate the social influences from different angles (topics). How to quantify the strength of those social influences. How to estimate the model on real large networks. To address these fundamental questions, we propose Topical Affinity Propagation (TAP) to model the topic-level social influence on large networks. In particular, TAP can take results of any topic modeling and the existing network structure to perform topic-level influence propagation. With the help of the influence analysis, we present several important applications on real data sets such as 1) what are the representative nodes on a given topic 2) how to identify the social influences of neighboring nodes on a particular node. To scale to real large networks, TAP is designed with efficient distributed learning algorithms that is implemented and tested under

the Map-Reduce framework. We further present the common characteristics of distributed learning algorithms for Map-Reduce. Social network analysis often focus on macro-level models such as degree distributions, diameter, clustering coefficient, communities, small world effect, preferential attachment, etc. work in this area includes Recently, social influence study has started to attract more attention due to many important applications. However, most of the works on this area present qualitative findings about social influences. In this paper, we focus on measuring the strength of topic-level social influence quantitatively. Finally, we demonstrate the effectiveness and efficiency of TAP on real large data sets. The goal of topic-based social influence analysis is to capture the following information: nodes' topic distributions, similarity between nodes, and network structure. In addition, the approach has to be able to scale up to a large scale network. Following this thread, we first propose a Topical Factor Graph (TFG) model to incorporate all the information into a unified probabilistic model. Second, we propose Topical Affinity Propagation (TAP) for model learning. Third, we discuss how to do distribute learning in the Map-Reduce framework. Finally, we illustrate several applications based on the results of social influence analysis.

U Kang, Charalampos E. Tsourakakis, Christos Faloutsos**.** In this paper, we describe PEGASUS, an open source Peta Graph Mining library which performs typical graph mining tasks such as computing the diameter of the graph, computing the radius of each node and finding the connected components. As the size of graphs reaches several Giga-, Tera- or Peta-bytes, the necessity for such a library grows too. To the best of our knowledge, PEGASUS is the first such library, implemented on the top of the H ADOOP platform, the open source version of M APREDUCE. Many graph mining operations (PageRank, spectral clustering, diameter estimation, connected components etc.) are essentially a repeated matrix-vector multiplication. Graphs are ubiquitous: computer networks, social networks, mobile call networks, the World Wide Web [1], protein regulation networks to name a few. The large volume of available data, the low cost of storage and the stunning success of online social networks and web2.0 applications all lead to graphs of unprecedented size. Typical graph mining algorithms silently assume that the graph fits in the memory of a typical workstation, or at least on a single disk; the above graphs violate these assumptions, spanning multiple Giga-bytes, and heading to Tera- and Peta-bytes of data. Unification of seemingly different graph mining tasks, via a generalization of matrix-vector multiplication; The careful implementation of GIM-V, with several optimizations, and several graph mining operations (PageRank, Random Walk with Restart(RWR), diameter estimation, and connected components). Moreover, the method is linear on the number of edges, and scales up well with the number of available machines; Performance analysis, pinpointing the most successful combination of optimizations, which lead to up to 5 times better speed than naive implementation.4) Analysis of large, real graphs, including one of the largest publicly available graph that was ever analyzed, Yahoo's web graph. Map Reduce and Hadoop.: MAPREDUCE is a programming framework for processing huge amounts of unstructured data in a massively parallel way. MAPREDUCE has two major advantages: (a) the programmer is oblivious of the details of the data distribution, replication, load balancing etc. and furthermore (b) the programming concept is familiar, i.e., the concept of functional programming. Briefly, the programmer needs to provide only two functions, a map and a reduce. The typical framework is as follows: (a) the map stage sequentially passes over the input file and outputs (key, value) pairs; (b) the shuffling stage groups of all values by key, (c) the reduce stage processes the values with the same key and outputs the final result. The proposed HCC, a new algorithm for finding connected components in large graphs. Like HADI, HCC is an application of GIM-V with custom functions.

U Kang Brendan Meeder Christos Faloutsos**.** Given a graph with billions of nodes and edges, how can we find patterns and anomalies? Are there nodes that participate in too many or too few triangles? Are there close-knit near-cliques? These questions are expensive to answer unless we have the first several eigenvalues and eigenvectors of the graph adjacency matrix. However, Eigen solvers suffer from subtle problems (e.g., convergence) for large sparse

matrices, let alone for billion-scale ones. We address this problem with the proposed HEIGEN algorithm, which we carefully design to be accurate, efficient, and able to run on the highly scalable MAPREDUCE (HADOOP) environment. This enables HEIGEN to handle matrices more than 1000 larger than those which can be analyzed by existing algorithms. We implement HEIGEN and run it on theM45 cluster, one of the top 50 supercomputers in the world. We report important discoveries about near-cliques and triangles on several real-world graphs, including a snapshot of the Twitter social network (38Gb, 2 billion edges) and the "Yahoo Web" dataset, one of the largest publicly available graphs (120Gb, 1.4 billion nodes, 6.6 billion edges). Graphs with billions of edges, or billion-scale graphs, are becoming common; Facebook boasts about 0.5 billion active users, who-calls-whom networks can reach similar sizes in large countries, and web crawls can easily reach billions of nodes. Given a billion scale graph, how can we find near-cliques, the count of triangles, and related graph properties? As we discuss later, triangle counting and related expensive operations can be computed quickly, provided we have the first several eigenvalues and eigenvectors. In general, spectral analysis is a fundamental tool not only for graph mining, but also for other areas of data mining. Eigenvalues and eigenvectors are at the heart of numerous algorithms such as triangle counting, singular value decomposition (SVD), spectral clustering, and tensor analysis. In spite of their importance, existing Eigen solvers do not scale well. In this paper, we discover patterns on near-cliques and triangles, on several realworld graphs including a Twitter dataset (38Gb, over 2 billion edges) and the "YahooWeb" dataset, one of the largest publicly available graphs (120Gb, 1.4 billion nodes,6.6 billion edges). To enable discoveries, we propose HEIGEN, an eigensolver for billion-scale, sparse symmetric matrices built on the top of HADOOP, an open-source MAPREDUCE framework. Our contributions are the following:

Siddharth Suri Sergei Vassilvitskii**.** The clustering coefficient of a node in a social network is a fundamental measure that quantifies how tightly-knit the community is around the node. Its computation can be reduced to counting the number of triangles incident on the particular node in the network. In case the graph is too big to fit into memory, this is a non-trivial task, and previous researchers showed how to estimate the clustering coefficient in this scenario. A different avenue of research is to perform the computation in parallel, spreading it across many machines. In recent years Map Reduce has emerged as a de facto programming paradigm for parallel computation on massive data sets. The main focus of this work is to give Map Reduce algorithms for counting triangles which we use to compute clustering coefficients. Author's contributions are twofold. First, we describe a sequential triangle counting algorithm and show how to adapt it to the MapReduce setting. This algorithm achieves a factor of 10-100 speed up over the naive approach. Second, we present a new algorithm designed specifically for the MapReduce framework. A key feature of this approach is that it allows for a smooth tradeoff between the memory available on each individual machine and the total memory available to the algorithm, while keeping the total work done constant. Moreover, this algorithm can use any triangle counting algorithm as a black box and distribute the computation across many machines. We validate our algorithms on real world datasets comprising of millions of nodes and over a billion edges. In recent years there has been an explosion in the amount of data available for social network analysis. It has become relatively common to study social networks consisting of tens of millions of nodes and billions of edges. Since the amount of data to be analyzed has grossly outpaced advances in the memory available on commodity hardware, computer scientists have once again turned to parallel algorithms for data processing.

Rasmus pagh and charalampos e. Tsourakakis**.** In this note we introduce a new randomized algorithm for counting triangles in graphs. We show that under mild conditions, the estimate of our algorithm is strongly concentrated around the true number of triangles. Specifically, if p >= max (log n /t,log n /Sqrt(t)), where n, t, denote the number of vertices in G, the number of triangles in G, the maximum number of triangles an edge of G is contained, then for any constant > 0 our unbiased estimate T is concentrated around its expectation, i.e., Pr [T- E [T] j >=E [T]] = o(1). Finally, we present a

Map Reduce implementation of our algorithm. Triangle counting is a fundamental algorithmic problem with many applications. The interested reader is urged to see [17] and references therein. The fastest exact triangle counting algorithm to date (in terms of number of edges in the graph) is due to Alon, Yuster and Zwick [3] and runs in O(m 2!!+1), where currently the matrix multiplication exponent! is 2.371 [9]. For planar graphs linear time algorithms are known, e.g., [18]. Practical methods for exact triangle counting use instead enumeration techniques, see e.g., [19] and references therein. For many applications, especially in the context of large social networks, an exact count is not crucial but rather a fast, high quality estimate. Most of the work on approximate triangle counting is sampling-based and has considered a (semi-)streaming setting. A different line of research is based on a linear algebraic approach. Currently to the best of our knowledge, the state-of-the-art approximate counting method relies on a hybrid algorithm that first sparsifies the graph and then samples triples according to a degree based partitioning trick.

Mainly, machine learning and data mining are used for Healthcare fraud detection. Supervised, unsupervised and semi-supervised learning are the three categories of Machine learning approaches. In most of the cases, semi-supervised learning approaches are used by many researchers. But, to detect frauds in healthcare system more efficiently, new semi supervised learning approaches can be proposed in few cases. But, to conceal all the instances of the healthcare fraud, there doesn't exist any particular standard approach or patterns. It can be concluded from this review that the advanced machine learning techniques and newly acquired sources of the healthcare data would be forthcoming subjects of interest in order to make the healthcare affordable, to improve the effectiveness of healthcare fraud detection and to bestow top quality on healthcare systems.

Hossein Joudaki, Arash Rashidian, Behrouz Minaei-Bidgoli, Mahmood Mahmoodi, Bijan Geraili, Mahdi Nasiri and Mohammad Arab**.** In this paper the authors stated that inappropriate payments by insurance organizations or third party payers occur because of errors, abuse and fraud. The scale of this problem is large enough to make it a priority issue for health systems. Traditional methods of detecting health care fraud and abuse are time-consuming and inefficient. Combining automated methods and statistical knowledge lead to the emergence of a new interdisciplinary branch of science that is named Knowledge Discovery from Databases (KDD). Data mining is a core of the KDD process. Data mining can help third-party payers such as health insurance organizations to extract useful information from thousands of claims and identify a smaller subset of the claims or claimants for further assessment. They reviewed studies that performed data mining techniques for detecting health care fraud and abuse, using supervised and unsupervised data mining approaches. Most available studies have focused on algorithmic data mining without an emphasis on or application to fraud detection efforts in the context of health service provision or health insurance policy. More studies are needed to connect sound and evidence-based diagnosis and treatment approaches toward fraudulent or abusive behaviors. Ultimately, based on available studies, we recommend seven general steps to data mining of health care claims.

They concluded that they presented a novel framework to deal with the so called flow of health data which, exploiting the strength of social networks, is able to provide a representation of the health care system as a whole allowing the clinicians to draw a comprehensive picture of the health state of patients. Their framework (both the data model and the analysis algorithms) was designed to be as intuitive as possible for users who less likely have a background in computer science. To this end, the analysis approach was inspired from the cognitive process of gaining awareness about a new phenomenon. Their system has shown to be able to provide effective answers to complex enquiries submitted by clinicians for which standard statistical methods fail.

## SYSTEM METHODOLOGY

### Graph Model

In this add node module, the node details such as Node Id, IP Address, X Position and Y Position are registered and the details are saved in 'Nodes' table. In edge from and to node id details are saved in 'Edges' table. Any number of nodes can be added as friend nodes to given

nodes. In view form the records from 'Nodes' are 'Edges' table are taken and the graph is displayed in Panel control graphically.

## Frequent Mining Process

In this module, mining frequent sub graph steps are carried out. Let G = {G1, G2,, Gn} be a graph database, where each Gi ∈G, ∀ I = {1... n} represents a labeled, undirected, simple (no multiple edges between a pair of vertices), and connected graph. For a graph g, its size is defined as the number of edges it contains. Now, t (g) = Gi : g ⊆ Gi ∈G, ∀ I = {1... n}, is the support-set of the graph g (here the subset symbol denotes a subgraph relation). Thus, t(g) contains all the graphs in that has a subgraph isomorphic to g. The cardinality of the support-set is called the support of g.g is called frequent if support $\geq \pi^{min}$, where $\pi^{min}$ is predefined/user-specified minimum support (minsup) threshold. The set of frequent patterns are represented by *F*. Based on the size (number of edges) of a frequent pattern, we can partition *F* into a several disjoint sets, *Fi* such that each of the *Fi* contains frequent patterns of size i only.

The mining is carried out using Iterative Map-Reduce approach. FSM-H is designed as an iterative MapReduce process. At the beginning of iteration i, FSM-H has at its disposal all the frequent patterns of size i – 1 (*Fi* - 1), and at the end of iteration i, it returns all the frequent patterns of size i, (*Fi*) . Note that, in this work, the size of a graph is equal to the number of edges it contains. For a mining task if is the set of fre-quent patterns, FSM-H runs for a total of l iterations, where l is equal to the size of the largest graph in *F*. This module implements an FSM algorithm that follows a typical candidate-generation-and-test paradigm with breadth-first candidate enumeration.

## Graph Representations

The least complex component whereby a diagram structure can be spoken to is by utilizing a contiguousness grid or nearness list. Utilizing a contiguousness network the lines and segments speak to vertexes, and the crossing point of line I and segment j speaks to a potential edge interfacing the vertexes vi and vj. The esteem held at convergence < I, j > regularly demonstrates the quantity of connections from vi to vj . Be that as it may, the utilization of nearness networks, albeit clear, does not fit isomorphism recognition, in light of the fact that a diagram can be spoken to from various perspectives relying uponWith respect to isomorphism testing it is therefore desirable to adopt a consistent labeling strategy that ensures that any two identical graphs are labeled in the same way regardless of the order in which vertexes and edges are presented (i.e. a canonical labeling strategy). A canonical labeling strategy defines a unique code for a given graph.

Canonical labeling facilitates isomorphism checking because it ensures that if a pair of graphs is isomorphic, then their canonical labeling will be identical. One simple way of generating a canonical labeling is to flatten the associated adjacency matrix by concatenating rows or columns to produce a code comprising a list of integers with a minimum (or maximum) lexicographical ordering imposed. To further reduce the computation resulting from the permutations of the matrix, canonical labeling are usually compressed, using what is known as a vertex invariant scheme that allows the content of an adjacency matrix to be partitioned according to the vertex labels. Various canonical labeling schemes have been proposed, some of the more significant are described in this subsection.

Canonical Adjacency Matrix (CAM): Given an adjacency matrix M of a graph g, an encoding of M can be obtained by the sequence obtained from concatenating the lower (or upper) triangular entries of M, including entries on the diagonal. Since different permutations of the set of vertexes correspond to different adjacency matrices, the canonical (CAM) form of g is defined as the maximal (or minimal) encoding. The adjacency matrix from which the canonical form is generated defines the Canonical Adjacency Matrix or CAM.

## Subgraph Enumeration

The current methods for enumerating all the subgraphs might be classified into two categories: one is the join operation adopted by FSG and AGM and another one is the extension operation. The major concerns for the join operation are that a single join might produce multiple candidates and that a candidate might be redundantly proposed by many join operations. The concern for the extension

operation is to restrict the nodes that a newly introduced edge may attach to. Equivalence class based extension is founded on a DFS-LS representation for trees. Basically, a (k + 1)-subtree is generated by joining two frequent k subtrees. The two k subtrees must be in the same equivalence class.

An equivalence class consists of the class prefix encoding, and a list of members. Each member of the class can be represented as a (l, p) pair, where l is the k-th vertex label and p is the depth-first position of the k-th vertex's parent. It is verified all potential (k + 1)-subtrees with the prefix [C] of size (k − 1) can be generated by joining each pair of members of the same equivalent class [C].Equivalence classes can be based on either prefix or suffix.

## Frequency Counting

Two Methods are used for graph counting: Embedding lists (EL) and Recomputed embeddings (RE). For graphs with a single node we store an embedding list of all occurrences of its label in the database. For other graphs a list is stored of embedding tuples that consist of (1) an index of an embedding tuple in the embedding list of the predecessor graph and (2) the identifier of a graph in the database and a node in that graph. The frequency of a structure is determined from the number of different graphs in its embedding list. Embedding lists are quick, but they do not scale very well to large databases. The other approach is based on maintaining a set of active" graphs in which occurrences are repeatedly recomputed.

## Frequent Sub-Graph Mining

Let, G = {G1, G2, . . . ,Gn} be a graph database, where each Gi ∈ G, ∀i = {1 . . . n} represents a labeled, undirected and connected graph. For a graph g, its size is defined as the number of edges it contains. Now, **t**(g) = {Gi : g ⊆ Gi ∈ G}, ∀i = {1 . . . n}, is the support-set of the graph g (here the subset symbol denotes a subgraph relation). Thus, **t**(g) contains all the graphs in G that has a subgraph isomorphic to g. The cardinality of the support-set is called the support of g. g is called frequent if support ≥ πmin, where πmin is predefined/user-specified minimum support (minsup) threshold. The set of frequent patterns are represented by F. Based on the size (number of edges) of a frequent pattern, we can partition F into a several disjoint sets, Fi such that each of the Fi contains frequent patterns of size i only.

## Mapreduce

MapReduce is a programming model that enables distributed computation over massive data. The model provides two abstract functions: map, and reduce. Map corresponds to the "map" function and reduce corresponds to the "fold" function in functional programming. Based on its role, a worker node in MapReduce is called a mapper or a reducer. A mapper takes a collection if (key, value) pairs and applies the map function on each of the pairs to generate an arbitrary number of (key,value) pairs as intermediate output. The reducer aggregates all the values that have the same key in a sorted list, and applies the reduce function on that list. It also writes the output to the output file

Iterative MapReduce: Iterative MapReduce can be defined as a multi staged execution of map and reduce function pair in a cyclic fashion, i.e. the output of the stage i reducers is used as an input of the stage i + 1 mappers. An external condition decides the termination of the job. Pseudo code for iterative MapReduce algorithm is presented in Figure.

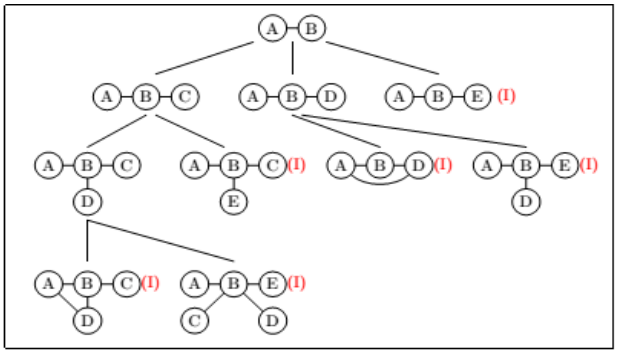


**Fig 3.1 Map Reduces Mining**

Algorithm: Iterative MapReduce():
While(Condition)
Execute MapReduce Job
Write result to DFS
Update condition
// G is the database
// k is initialize to 1
Mining Frequent Subgraph(G, minsup):
Populate F1
while Fk 6= ∅
Ck+1 = Candidate generation(Fk, G)
forall c ∈ Ck+1
if isomorphism checking(c) = true
support counting(c, G)
if c.sup ≥ minsup
Fk+1 = Fk+1 S{c} k = k + 1
return Si=1•••k−1 Fi

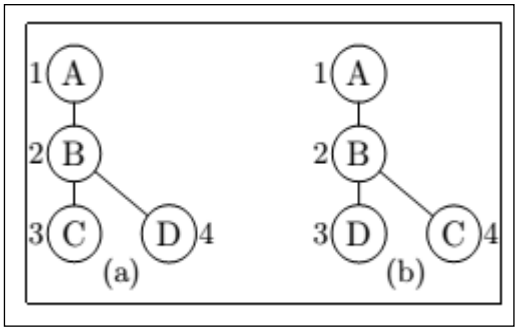
In this paradigm, the mining task starts with frequent patterns of size one (single edge patterns), denoted as F1 (Line 0). Then in each of the iterations of the while loop (Line 1-6), the method progressively finds F2, F3 and so on until the entire frequent pattern set (F) is obtained. If Fk is non-empty at the end of an iteration of the above while loop, from each of the frequent patterns in Fk the mining method creates possible candidate frequent patterns of size k+1 (Line 2). These candidate patterns are represented as the set C. For each of the candidate patterns, the mining method computes the pattern's support against the dataset G (Line 5). If the support is higher than the minimum support threshold (minsup), the given pattern is frequent, and is stored in the set Fk+1 (Line 6). Before support counting, the method also ensures that different isomorphic forms of a unique candidate patterns are unified and only one such copy is processed by the algorithm (Line 4). Once all the frequent patterns of size k + 1 are obtained, the while loop in Line 1 to 7 continues.

Thus each iteration of the while loop obtains the set of frequent patterns of a fixed size, and the process continues until all the frequent patterns are obtained. In Line 8, the FSM algorithm returns the union of Fi : $1 \leq i \leq k - 1$.

## ISOMORPHISM CHECKING

Given a frequent pattern of size k, this step adjoins a frequent edge (which belongs to F1) with c to obtain a candidate pattern d of size k + 1. If d contains an additional vertex then the added edge is called a forward edge, otherwise it is called a back edge; the latter simply connects two of the existing vertices of c. Additional vertex of a forward edge is given an integer id, which is the largest integer id following the ids of the existing vertices of c; thus the vertex-id stands for the order in which the forward edges are adjoined while building a candidate pattern. In graph mining terminology, c is called the parent of d, and d is a child of c, and based on this parent-child relationship we can arrange the set of candidate patterns of a mining task in a candidate generation tree.

FSM algorithms also impose restriction on the extension nodes of the parent pattern so that redundant generation paths can be reduced. One such restriction that is used in the popular gSpan algorithm is called rightmost path generation that allows adjoining edges only with vertices on the rightmost path. Simple put, "right most vertex" (RMV) is the vertex with the largest id in a candidate subgraph and "right most path" (RMP) is the shortest path from the lowest id vertex to the RMV strictly following forward edges.




| Pattern | Occerence List (OL) |
|---|---|
| A–B | 1 : [(1, 2)] ; 2 : [(1, 2)] |
| B–D | 1 : [(2, 4)] ; 2 : [(2, 3)] ; 3 : [(1, 2)] |
| B–E | 2 : [(2, 5)] ; 3 : [(1, 3)] |
| A–B–D | 1 : [(1, 2), (2, 4)] ; 2 : [(1, 2), (2, 3)] |
| A–B–E | 2 : [(1, 2), (2, 5)] |

**Fig 3.2 Isomorphism Graph**

Mention the previous paragraph, a candidate pattern can be generated from multiple generation paths, but only one such path is explored during the candidate generation step and the remaining paths are identified and subsequently ignored. To identify invalid candidate generation paths, a graph mining algorithm needs to solve the graph isomorphism task, as the duplicate copies of a candidate patterns are isomorphic to each other. A well-known method for identifying graph isomorphism is to use canonical coding scheme, which serializes the edges of a graph using a prescribed order and generates a string such that all isomorphic graphs will generate the same string. There are many different canonical coding schemes, min- dfs-code is one of those which is used in .

Mapper FSG(Fkphx.min-dfs-code, x.obji):
Ck+1 = Candidate generation(Fkp)
forall c ∈ Ck+1
if isomorphism checking(c) = true
populate occurrence List(c)
if length(c.occurrence List) > 0
emit (c.min-dfs-code , c.obj)

In this study present a brief overview of the current status and future directions of frequent pattern mining. There are various inter-disciplinary domains like chemo informatics, bioinformatics etc. where mining of recurrent patterns across large collection of networks is required. Due to increasing size and complexity of patterns in there is a need for efficient graph mining algorithm. With over a decade of extensive research, there have been hundreds of research publications and tremendous research, development and application activities in this domain.

Many algorithms for frequent subgraph mining have been proposed so far. Most of the algorithms, they focus only on a static set of graphs. Very few algorithms are for mining patterns from dynamic set of graphs. Also all the algorithms proposed so far, outperform each other, either in terms of memory requirements or in terms of few orders of magnitude of computation time. None of them completely address the issue of NP-completeness of the subgraph mining problem.

## CONCLUSION

This paper proposes HNCDPM to help detect health insurance fraud. The developed method helps us understand the progression of chronic disease, including orphan diseases, and is helpful in detecting chronic disease-related fraud and reducing healthcare costs. HNCDPM considers different medication periods of the same disease and produces two types of rules: the pattern between different stages of different chronic diseases, which indicates the relationship between different types of chronic disease, and the pattern between different stages of the same chronic disease, which shows the clinical path of the disease. These two types of rules can be used to help detect chronic disease fraud. The proposed system presented a novel iterative Map Reduce based frequent sub graph mining algorithm, called FSM-H. The proposed system shows the performance of FSM-H over real life and large synthetic datasets for various system and input configurations. In this project also compare the execution time of FSM-H with an existing method, which shows that FSM-H is significantly better than the existing method.

## REFERENCES

[1]. G. Liu,M. Zhang, and F. Yan, "Large-scale social network analysis based on Mapreduce," in Proc. Int. Conf. Comput. Aspects Soc. Netw., 2010, 487–490.

[2]. J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," Commun. ACM, 51, 2008, 107–113.

[3]. U. Kang, C. E. Tsourakakis, and C. Faloutsos, "Pegasus: A petascale graph mining system implementation and observations," in Proc. 9th IEEE Int. Conf. Data Mining, 2009, 229–238.

[4]. U. Kang, B. Meeder, and C. Faloutsos, "Spectral analysis for billion-scale graphs: Discoveries and implementation," in Proc.15th Pacific-Asia Conf. Adv. Knowl. Discov. Data Mining, 2011, 13–25.

[5]. S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," in Proc. 20th Int. Conf. World Wide Web, 2011, 607–614.

[6]. R. Pagh and C. E. Tsourakakis, "Colorful triangle counting and a mapreduce implementation," Inf. Process. Lett., 112(7), 2012, 277–281.

[7]. F. Afrati, D. Fotakis, and J. Ullman, "Enumerating subgraph instances using map-reduce," in Proc. IEEE 29, 2013, 62–73.

[8]. B. Bahmani, R. Kumar, and S. Vassilvitskii, "Densest subgraph in streaming and mapreduce," Proc. Very Large Data Bases Endow., 5(5), 454–465, 2012.

[9]. J. Huan, W. Wang, D. Bandyopadhyay, J. Snoeyink, and J. Prins, "Mining protein family specific residue packing patterns from protein structure graphs," in Proc. Int. Conf. Res. Comput. Mol. Biol., 2004, 308–315

[10]. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in Proc. Conf. Very Large Data Bases, 20, 1994.