IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 3, Issue 7, July 2016

Unfolding and Boosting based Graph Partitioning Approach for Object-Oriented System

Appala Srinuvasu Muttipati[1], Padmaja Poosapati[2]

Research Scholar, Department of Computer Science and Engineering, GITAM University, Visakhapatnam, India[1]

Associate Professor, Department of Information Technology, GITAM University, Visakhapatnam, India[2]

Abstract: Object oriented design is a template to solve a problem that can be used in many different applications. For the design of a new application, a large amount of code can be reused by the existing design patterns. Therefore, it becomes necessary to identify the similar design structures. One of the frequently used algorithms for finding identical structures is Markov Clustering (MCL), that works well when the clusters are small in diameter, but for large clusters, there is a possibility that the weakly connected components may overlap. To remove overlapping, authors adopted and modified the existing MCL approach and thus proposed the new graph partitioning approach named as Unfolding and Boosting-based Graph Partitioning Algorithm (UB-GPA), that will result in an inconsistent set of uni-class clusters. The algorithm interprets the clusters/partitions by an iterative process of unfolding and boosting operations. Experimental results show that the proposed UB-GPA performs well as compared with MCL.

Keywords: software design models; reusability; identical design structures; software clustering; graph partitioning.

I. INTRODUCTION

Software systems are hard to comprehend the functionality of the large systems because systems are implemented with concepts of object oriented programming. Software systems are more difficult to extend and modify them. So, large and complex software systems tend to break into small components that are easier to understand by a software developer or designer.

The concept of breaking a large software system into small components can be achieved through clustering. Clustering is a useful and important unsupervised learning technique widely studied in the literature [1-3].The universal goal of clustering is to group similar objects into one cluster while partitioning dissimilar objects into different clusters. Clustering has broad applications including the analysis of biological data, financial data, time series data and spatial data soon.

Graph as an expressive data structure is widely used to model a structural relationship between objects in many application domains such as the web, social networks, biological networks and fraud detection, etc. Graph clustering is an interesting and challenging research problem which has received much attention recently [4] Clustering on a large graph aims to partition the graph into several densely connected components. Typical applications of graph clustering include community detection in social networks, identification of functionally related protein modules in large protein-protein interaction networks, etc [5].

Software clustering is an imperative facility to decompose a current system into smaller parts. It serves to identify the subsystems that are related functionally and are someway independent of the other part of the system. Therefore, the procedure of reverse engineering a software system must identify the clusters that represent the system. The usual reason for reverse engineering a piece of software is to restructure the program, to build something similar to it, to exploit its weaknesses or strengthen its defences.

Object oriented domain an essential principal, which is also a golden rule in designing reusable software, is that of modularity. A module in a software system is a single unit of the application design. Each module is embodied with a small number of classes that are strongly coupled. Modules must have a simple interface through which they can interrelate with each other modules. Likewise, the coupling between modules should be loose in order to maintain their autonomy.

In this paper, we propose a new graph partitioning approach for partitioning an object-oriented software system. By considering a few features of Markov clustering, a new graph partitioning approach is implemented. Markov clustering is successfully applied in the fields of bioinformatics, biological networks, community detection in graphs etc, where vertices form groups with a higher density of edge weights within a group and lower density edge weights between the groups. As the static software model graph of object-oriented software display small-world behavior and community structure, authors proposed the application of new graph partitioning algorithm for software clustering. The algorithm can work on undirected weighted graphs and it is observed that the new algorithm considerable advantage over spectral partitioning and Markov Clustering (MLC) algorithms. In our approach, we first create the undirected weighted graph of software, where vertices graph is classes, abstracts and interfaces, and the edge represents relations between the entities. Then we apply the Unfolding and Boosting based Graph Partitioning algorithm (UB-GPA) to find partitions in software. The

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 3, Issue 7, July 2016

approach is evaluated on three hypothetical software systems from [6-7]. We compared the results obtained by proposed approach with existing spectral partitioning algorithm is found to be same and UB-GPA achieved a substantial performance over object oriented software clustering.

II. RELATED WORK

In this related work, describes the graph clustering methods and how the graph clustering techniques are playing a role in software engineering and different data transformation techniques.

The graph clustering is mainly done in two ways first one is between graphs and another one is within a graph. Between a graph clustering approaches split a set of graphs into various clusters. For example, chemical compounds structures can be assembled into clusters based on their structural similarity. Within a graph, clustering methods split the nodes of a graph into clusters. For example, in social networking graphs can be clustered based on their similar activities or institutions etc.

This literature gives different algorithms to perform within-graph clustering methods such as a k-Spanning tree, Shared Nearest Neighbor, Betweenness Centrality Based, Highly Connected Components, Maximal Clique Enumerations, Kernel k-means, Spectral partitions and Markov Clustering.

Several techniques and approaches for software clustering are introduced in the study. Shtern et al. presented a review article on clustering methodologies for software engineering. Authors described each phase of clustering algorithms separately and important approaches for evaluating the effectiveness of software clustering [8].

Hierarchical clustering algorithms are two categories: agglomerative is a bottom-up approach and divisive is a top-down approach. The agglomerative algorithm starts from the bottom of the hierarchy by iteratively grouping similar entities into clusters. At each step, the two clusters that are most similar to each other are merged, and the number of clusters is reduced by one. Divisive algorithms start with one cluster that contains all entities and divide the cluster into a number (usually two) of separate clusters at each successive step.

Czibula et al. presented a new hierarchical agglomerative clustering algorithm utilized for object-oriented software systems restructuring [9]. This methodology aims at identifying a partition of a software system that corresponds to an improved structure of it. Hierarchical Agglomerative clustering algorithm for Restructuring Software systems (HARS) can be used in the grouping step of Clustering Approach for Refactoring Determination (CARD) in order to re-group entities from the software system. This methodology can be useful for assisting software engineers in their daily works of refactoring software systems. They evaluated using the open source JHotDraw (i.e. a java GUI framework for technical and structured graphics) emphasizing its advantages in comparison with existing approaches.

Spectral graph partitioning methods first appeared in the early seventies in research work of [10-12]. They explored the properties of the algebraic representations of the graphs and introduced the idea of using eigenvectors for partitioning the graph. These methods have been applied to many research disciplines, related to computer science and electrical engineering. Hendrickson et al., [13] used Spectral Graph Partitioning in parallel systems. Pothen et al. [14] and Bernard et al. [15] used similar techniques in scientific computing. This Spectral partitioning is still using and providing better partitions, but it having somewhat complex calculations is required to perform partitions.

Xanthos proposed a spectral graph partitioning method to decompose the object-oriented software system [6]. The methodology is based on an iterative process for partitioning the graph in order to identify dense communities of classes thereby minimizing the communication between the modules of the system. This is achieved because it finds the minimum cut set of edges in each iteration. Therefore, it can also be utilized to identify the modules that should be assigned in diverse machines, in a distributed environment.

Dongen suggested a Markov Cluster (MCL) Algorithm [16] that involves changing the values of a transition matrix towards either 0 or 1 at every progression in a random walk until the stochastic conditions are satisfied. When the Hadamard power [17] for each transition probability value is divided by the sum of each column, the rescaling procedure yields a transition matrix for the next stage. After repeating alternatively for around 20 times between two stages random walk and probability modification, the procedure will, at last, achieve a convergence stage in which the whole graph is subdivided into a set of 'hard' clusters. The benefits of the algorithm are (i) it is not misled by edges linking different clusters; (ii) it scales well with increasing graph size; (iii) it has a natural parameter for influencing cluster granularity. The limitations of the algorithm are (i) it cannot find overlapping clusters (ii) it is not suitable for clusters with a large diameter.

Han et al. described normalization which is one of the data transformation methods in data preprocessing [18]. An attribute of a dataset is normalized by scaling its value so that they fall within a small range, such as 0.0 to 1.0. Normalization may improve the accuracy and efficiency of mining algorithms in classification and clustering. There are numerous methods for data normalization that include min-max normalization, z-score normalization and normalization by decimal scaling.

III.PROPOSED APPROACH

The proposed approach consists of two phases. In the first phase complete graph model of the software design is built by extracting software entities (i.e. classes, abstracts, and interfaces) and relations directly from source code. The model graph obtained is a directed weighted graph. These graphs are regenerated as undirected weighted graph by

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 3, Issue 7, July 2016

removing the directions of the edges. In second phase software clusters are generated from the graph produced in the first phase. Authors have suggested a new graph partitioning algorithm by adopting few features of MCL algorithm. In this algorithm symmetric matrix M is formed by combining adjacency matrix and degree matrix of G. The Min-Max normalization technique is applied to matrix M for transforming the data to a specified small range. The algorithm consists of two main operations namely unfolding and boosting which are implemented on the normalized matrix. These operations are repeated alternatively until stable steady matrix is reached. Finally the clusters are interpreted from stable matrix S (i, j), where $1 \leq i \leq n$, $1 \leq j \leq n$, and n is number of vertices. For the rows having the value one, the corresponding j indices are taken into one group called cluster. Thus in this way the clustering are interpreted from the stable matrix. Fig. 1 shows the block diagram of the proposed approach UB-GPA and Fig. 2 depicts the UB-GPA flow.

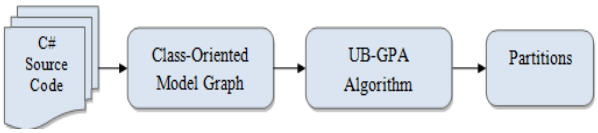

Fig.1 Block Diagram for Proposed approach UB-GPA

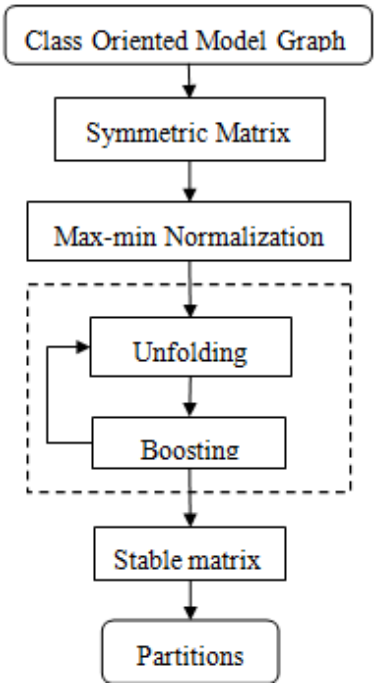

Fig.2 Flow of the UB-GPA Algorithm

A. Phase 1: Developing of Class-Oriented Model Graph

The developed software model graph is a simple directed weighted graph. Essential information such as system entities and associations are directly extracted from C# source code by parsing the abstract syntax [19]. The vertices of the generated graph are classes, abstract and interface. The edges are the associations between the entities. Association types in the graph are based on associations of UML class diagrams. Normally there can be more than one association between entities (classes, abstracts, interfaces). Therefore to create a simple graph all the parallel edges are merged into one single edge. The feasible association types between classes are (i) Extend association X, where Class B is the base class of Class A, (ii) Implement association I, where Class A implements the interface of class B, (iii) Field type association A, where Class A has a field type of Class B, (iv) Local variable association L, where Class A method has a local parameter with the type of Class B, (v) Parameter association P, where Class A method has an input parameter with the type of Class B, (vi) Return type association R, where Class A has a method with the return type of Class B, (vii) Method association M, where Class A has a method calls to Class B.

An object-oriented system can be represented as a digraph G (V, E, L_v, L_e, v, e) where V is a set of vertices (classes), E is a set of edges (associations), Lv is a set of labels for vertices, L_e is a set of labels for edges. v is a mapping from $V \rightarrow L_v$, e is a mapping from $E \rightarrow L_e$ Assigning weights to edges according to their strengths of the associations will improve the performance of the approach and enhance the quality of the detection results by reducing the ratio of the false positive detections.

The obtained software model graph is directed weighted graph. In this paper, authors propose a new graph partitioning algorithm approach that supports undirected graphs. So, the obtained directed weighted graph is regenerated as the undirected weighted graph by removing the directions of the edges. Fig.3 shows the creation of undirected weighted software model graph with parallel edges and without parallel edges (i.e. if any subgraph contain X→Y and Y→ X then it considered as the X→Y and combining their edge weights).

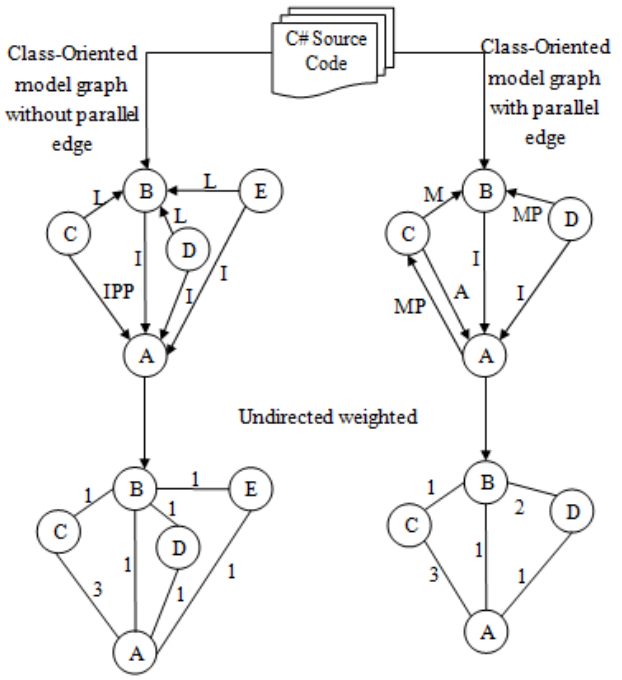

Fig.3 Creation of undirected weighted software model graph

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 3, Issue 7, July 2016

B. Phase 2: Generating clusters by unfolding and boosting operations

The input for the second phase is an undirected weighted graph G, unfold parameter 'm' and boost parameter 'b'. From the input graph G, an adjacency matrix A is created and is represented as A= [a_{ij}], where a_{ij} is the weight of the edge e_{ij}. Degree matrix D of G is found, which is represented as D= [d_{ij}] and is defined as

$$d_{ij} = \begin{cases} \sum_{k=1}^{n} a_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

From adjacency matrix not possible to find proper clusters, because the node value (i.e. M(1, 1), M(2, 2), ..) are zero in the matrix. Degree matrix provides information of the node which helps to get proper clusters. Adjacency and Degree matrix of G are combined to obtain the symmetric matrix M.

The resultant symmetric matrix M is normalized by Min-max normalization technique for transforming the data to a specified small range. The Min-max normalization maps a value v to v' in the range $[new\_min_A, new\_max_A]$ for the attribute A, by computing

$$v' = \frac{P}{Q} * [new\_max_A - new\_min_A] + new\_min_A \quad (2)$$

Where P= [v-minimum value of A] and Q= [maximum value of A-minimum value of A]

In the proposed approach the minimum and the maximum value of the column of a matrix M is taken and the new range for the matrix is defined from [0, 1]. Thus the above equation for the matrix M (i, j) can be written as

$$v' = \frac{X}{Y} * (new\_range\_max - new\_range\_min) + new\_range\_min \quad (3)$$

Where X = [v-min value of jth column], Y= [max value of jth column – min value of jth column] and 1≤ j≤ n, n is number of vertices

After normalizing the matrix two main operations, namely unfolding and boosting are applied. These operations are repeated iteratively until a convergence is reached i.e. until a stable matrix is reached.

The Unfolding matrix M_U is achieved via matrix squaring and is defined as

$$M_U = M'_{ij} = \sum_{k=1}^{m} M_{ik} M_{kj} \quad (4)$$

Where, m is an integer taken as 2. Clusters of large size are formed with the higher value of m. This may lead to a single cluster with no partitions and lower value of m (i.e. m=1) may lead to the problem of multiplication. Therefore m value is considered as 2.

The boosting is achieved by element square of the unfold matrix M' and element squared matrix is normalized. The boosting operation on the present state is responsible for both strengthening due to element squaring and weakening due to normalizing. Therefore to control the extent of the strengthening / weakening, boosting parameter b is considered. The low boosting parameter is more prone to yield smooth partitions.

$$M_B = M''_{ij} = \frac{(M'_{ij})^b}{normalization} \quad (5)$$

Where, b is an integer value taken as 2.

The unfolding and boosting operations are iteratively repeated until it results in a stable state matrix. The stable matrix is identified by comparing the result matrix obtained and the previously result obtained are to be same then it says that stable matrix is arise. Finally the clusters are interpreted from stable matrix S (i, j), where 1≤ i ≤ n and 1≤ j ≤ n, n is a number of vertices. For the rows having the value one, the corresponding j indices are taken into one group called a cluster. Thus in this way the clusters are interpreted from the stable matrix. The key benefits of the proposed UB-GPA algorithm are (i) Its formulation is simple and elegant. (ii) It needs less < 20 number of iteration to produce a stable state. (iii) It produces good clustering results and (iv) this approach is applicable to detect the community structures in the dense network. Illustration of the proposed UB-GPA algorithm: Let us consider an undirected weighted graph shown in Fig.4, which is an input for UB-GPA algorithm.

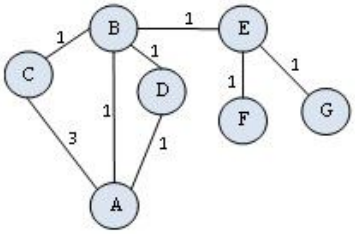

Fig.4. Undirected weighted graph

Initially an adjacency matrix A and degree matrix D for the input graph are created. The symmetric matrix M is obtained by combining an adjacency matrix A and Degree matrix D as shown below

$$A = \begin{bmatrix} 0 & 1 & 3 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 3, Issue 7, July 2016

$$M=\begin{bmatrix}5&1&3&1&0&0&0\\1&4&1&1&1&0&0\\3&1&4&0&0&0&0\\1&1&0&2&0&0&0\\0&1&0&0&3&1&1\\0&0&0&0&1&1&0\\0&0&0&0&1&0&1\end{bmatrix}$$

The obtained matrix M is n× n symmetric matrix, which is normalized within the new range between [0, 1] by using min -max normalization method. The min max normalization computation is done by using equation 3.

From the matrix M, first column elements are [5, 1, 3, 1, 0, 0, 0], where the maximum value is 3 and minimum value is 0. The value of M (1, 1) i.e. 3 is normalized to ((5-0) / (5-0)) * (1-0) + 0 = 1 and the value of M (2, 1) i.e. 1 is normalized to ((1-0) / (5-0)) * (1-0) + 0 = 0.33.

Similarly computing for entire matrix, normalized matrix M_N is

$$\begin{bmatrix}1.00&0.25&0.75&050&0.00&0.00&0.00\\0.20&1.00&0.25&0.50&0.33&0.00&0.00\\0.60&0.25&1.00&0.00&0.00&0.00&0.00\\0.20&0.25&0.00&1.00&0.00&0.00&0.00\\0.00&0.25&0.00&0.00&1.00&1.00&1.00\\0.00&0.00&0.00&0.00&0.33&1.00&0.00\\0.00&0.00&0.00&0.00&0.33&0.00&1.00\end{bmatrix}$$

On normalized matrix, unfold and boosting operations are applied. The unfold operation is a simple matrix squaring (i.e. $M_N \times M_N$). Thus the unfold matrix M_U is

$$\begin{bmatrix}1.60&0.81&1.56&1.13&0.08&0.00&0.00\\0.65&1.32&0.65&1.10&0.67&0.33&0.33\\1.25&0.65&1.51&0.43&0.08&0.00&0.00\\0.45&0.55&0.21&1.23&0.08&0.00&0.00\\0.05&0.50&0.06&0.13&1.75&2.00&2.00\\0.00&0.08&0.00&0.00&0.67&1.33&0.33\\0.00&0.08&0.00&0.00&0.67&0.33&1.33\end{bmatrix}$$

The boosting operation is the matrix element squaring, illustrating by taking the square of the first column element of M_U, that results with the values (5.19, 1.36, 3.16, 0.60, 0.01, 0, 0). Squaring is done for the remaining columns of matrix M_U resulting in a matrix as shown below.

$$\begin{bmatrix}2.56&0.66&2.44&1.27&0.01&0.00&0.00\\0.42&1.74&0.42&1.21&0.44&0.11&0.11\\1.56&0.42&2.29&0.18&0.01&0.00&0.00\\0.20&0.30&0.05&1.50&0.01&0.00&0.00\\0.00&0.25&0.00&0.02&3.06&4.00&4.00\\0.00&0.01&0.00&0.00&0.44&1.78&0.11\\0.00&0.01&0.00&0.00&0.44&0.11&1.78\end{bmatrix}$$

Boosting matrix M_B is obtained by using again the same min-max normalization method. Thus the resultant Boosting Matrix M_B is

$$\begin{bmatrix}1.00&0.38&1.00&0.84&0.00&0.00&0.00\\0.16&1.00&0.17&0.81&0.14&0.03&0.03\\0.61&0.24&0.94&0.12&0.00&0.00&0.00\\0.08&0.17&0.02&1.00&0.00&0.00&0.00\\0.00&0.14&0.00&0.01&1.00&1.00&1.00\\0.00&0.00&0.00&0.00&0.14&0.44&0.03\\0.00&0.00&0.00&0.00&0.14&0.03&0.44\end{bmatrix}$$

The unfolding and boosting operations are repeated alternatively on matrix M_B until the matrix gets a stable state. Finally a stable matrix M_S is obtained as

$$M_S=\begin{array}{c}\begin{matrix}A&B&C&D&E&F&G\end{matrix}\\\begin{bmatrix}1&1&1&1&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&1&1&1\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}\end{array}$$

From the matrix M_S the partitions are interpreted by finding the number of ones in a row and combining the corresponding indices into one group. Vertices A, B, C, D are considered as one group or one partition, since all the ones in the first row of M_S correspond to the indices A, B, C, D. Similarly vertices E, F, G belong to another group forming partition 2 which was depicted in Figure 5.

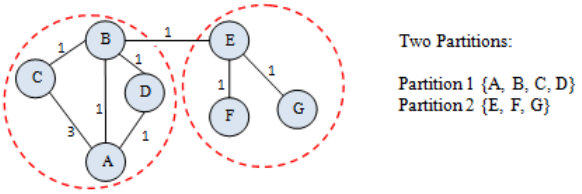

Fig. 5 Interpreted graph partitions from stable matrix

Algorithm: Unfolding and Boosting Graph Partitioning Algorithm (UB-GPA)

Input: Software model graph G, unfolding parameter '*m*', boosting parameter '*b*'
Output: Software partitions

Step 1: Create an adjacency matrix A of G
Step 2: Find the degree of the matrix D of G
Step 3: Combine the adjacency and degree matrix to form a symmetric matrix M i.e. M = D+A
Step 4: The symmetric Matrix M is normalized M_N using min-max normalization technique.
Step 5: Apply simple matrix multiplication $M_N \times M_N$ to obtain unfolding matrix M_U.
Step 6: Apply element squaring for obtain unfolding matrix and min-max normalization for the element squared matrix to obtain boosting matrix M_B.
Step 7: Repeat step 4 and step 5 until a stable state matrix MS is obtained.
Step 8: Finding the number of ones in a row and combine all the indices into one group to form a cluster from stable matrix generated in step 7.

DOI 10.17148/IARJSET.2016.3724

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 3, Issue 7, July 2016

IV. RESULTS AND DISCUSSION

In the result analysis, three different software model graphs are considered (Chatzigeorgiou et al., 2006; Xanthos, 2006). Table 1 gives the descriptive information about experimented software systems. In particular, first column shows different software model graphs. The second and third columns show the number of vertices and edges in the undirected graph (i.e. software model graph).

TABLE I DESCRIPTION OF THE SOFTWARE MODEL GRAPH

Software graph data	|V|	|E|
Software model graph 1	10	10
Software model graph 2	12	12
Software model graph 3	12	17

For each software model graph, the experimental analysis is done with three different approaches namely proposed UB-GPA, MCL Algorithm and spectral partitioning algorithm. Spectral partitioning algorithm is used in the experimental analysis of UB-GPA to show that the results obtain by the proposed UB-GPA are same as obtained by Spectral partitioning algorithm.

Fig.3 illustrates an example for UB-GPA, it takes 12 iterations to get stable state matrix, whereas MCL algorithm requires 9 iterations. MCL require less iteration count but having an overlapping of clusters. The proposed approach needs 3 more iterations to get stable matrix when comparing with the MCL. Here spectral partitioning algorithm is considered for justification of the obtained output is true for the UB-GPA algorithm.

Results of software model graph 1:
It is observed that software model graph 1 is partitioned into three clusters with the disconnected edges are 7-3 with weight 2 and 7-8 with weight 3.

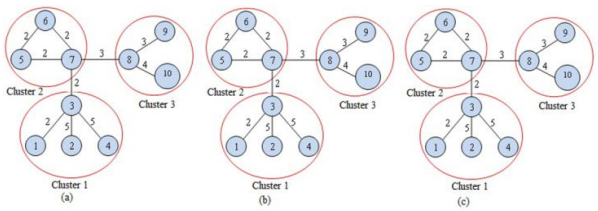

Fig.5 Partitions of software model graph 1 a) UB-GPA Algorithm b) MCL Algorithm c) Spectral partitioning algorithm

There is no overlapping of clusters in all the three algorithms. The results of software model graph 1 are shown in Fig.5.

Results of software model graph 2:
It is observed that software model graph 2 is partitioned into three clusters with the disconnected edges are 1-9 with weight 2 , 1-3 with weight 3 and 1-5 with weight 1. There is no overlapping of clusters in UB-GPA algorithm and Spectral partitioning algorithm. In MCL algorithm there is an overlapping of cluster1 and cluster 3 with vertex 9. The result of software model graph 2 is shown in Fig.6.

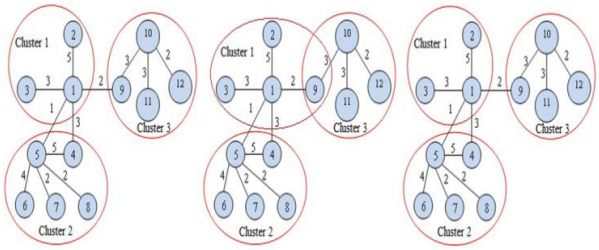

Fig.6 Partitions of software model graph 2 a) UB-GPA Algorithm b) MCL Algorithm c) Spectral partitioning algorithm

Results of software model graph 3:
It is observed that software model graph 3 is partitioned into two clusters with the disconnected edges are 6-7 with weight 1 and 2-7 with weight 1. There is no overlapping of clusters in UB-GPA algorithm and Spectral partitioning algorithm. In MCL algorithm there is an overlapping of cluster1 and cluster 2 with vertices 7, 8 and 9. The result of software model graph 3 is shown in Fig.7. The summary table can be viewed in Table II.

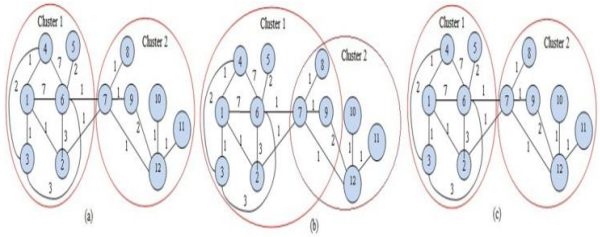

Fig.7 Partitions of software model graph 3 a) UB-GPA Algorithm b) MCL Algorithm c) Spectral partitioning algorithm

TABLE III SUMMARY TABLE OF THE RESULT

Hypothetical graph data	UB-GPA Algorithm		MCL Algorithm		Spectral partitioning Algorithm	
	Partitions	Over Lapping	Partitions	Over lapping	Partitions	Over lapping
Software model graph 1	P 1 {1, 2, 3,4 } P 2 {5,6,7} P 3 {8,9,10 }	No	P 1 {1, 2, 3,4 } P 2 {5,6,7} P 3 {8,9,10 }	No	P 1 {1, 2, 3,4 } P 2 {5,6,7} P 3 {8,9,10 }	No

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 3, Issue 7, July 2016

Software model graph 2	P 1{1,2,3} P 2 {4,5,6,7,8 } P 3 {9,10,11,12 }	No	P 1 {1, 2, 3, 9 } P 2 {4,5,6,7,8 } P 3 {9,10,11,12 }	Yes	P 1{1,2,3 } P 2{4,5,6,7,8 } P 3 {9,10,11,12 }	No
Software model graph 3	P 1 {1,2,3,4,5,6 } P2 { 7,8,9,10,11, 12}	No	P 1 {1, 2, 3, 4, 5, 6, 7, 8 ,9 } P 2 { 7,8,9,10,11, 12}	Yes	P 1 {1,2,3,4,5,6 } P2 { 7, 8, 9,10, 11, 12}	No

V. CONCLUSION

In this paper, a new graph partitioning approach is suggested namely Unfolding and Boosting- based Graph Partitioning algorithm (UB-GPA). The main objective is to implement UB-GPA algorithm to provide the best partitions for an object oriented system. This can be achieved by two main operations, one is simple matrix multiplication is called unfolding and another one is squaring the elements of the unfold matrix and normalizing the element squared matrix to the range [0, 1]. These operations are repeated alternatively to achieve a stable matrix state to form clusters. The suggested UB-GPA algorithm was examined on three different software model graphs. The stable matrix is occurred with low iterations while comparing with MCL algorithm iterations and it is observed that the suggested UB-GPA algorithm does not produce any overlapping groups/partitions. The results obtained were compared with existing spectral partitioning algorithm and are found to be same. Authors would like to come up with real time applications and find an optimal solution for software systems.

The outcome of UB-GPA algorithm can have multiple uses. 1) The partitions that are found can be utilized as reverse engineering process of the software system. 2) The partitions can be used for identification of design patterns. 3) The identified design patterns are transfer easily and utilized it in another system (reusability). 4) The partitions can show some design structures like common structures, copy past structures and circular dependency structures.

REFERENCES

[1] R.. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," In Proceedings of the ACM-SIGMOD International Conference Management of Data, Ashutosh Tiwary and Michael Franklin (Eds.), ACM, New York, NY, USA, 1998, pp.94-105.

[2] D. Gibson, J. Kleinberg, and P. Raghavan, "Inferring web communities from link topology," In Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space---structure in hypermedia systems: links, objects, time and space---structure in hypermedia systems (HYPERTEXT). ACM, New York, NY, USA, 1998; pp. 225-234.

[3] R. T. Ng and J. Han, "Efficient and effective clustering method for spatial data mining," In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994; pp. 144-155.

[4] D. A. Bader, H. Meyerhenke, P. Sanders, D.Wagner, "Graph Partitioning and Graph Clustering," 10th DIMACS Implementation Challenge Workshop February 13-14, 2012, Georgia Institute of Technology, Atlanta, GA. Contemporary Mathematics 588, American Mathematical Society and Center for Discrete Mathematics and Theoretical Computer Science, 2013.

[5] S. E. Schaeffer, "Graph Clustering," Computer Science Review, 2007, pp. 27-64.

[6] S. Xanthos, "Clustering Object-Oriented Software Systems using Spectral Graph Partitioning," In Proccedings of ACM Student Research Competition, Grand Finals, Second Award. 2005.

[7] A. Chatzigeorgiou, N.Tsantalis and Stephanides, "Application of Graph Theory to OO Software Engineering," In Proceedings of international workshop on workshop on interdisciplinary software engineering research, ACM New York, NY, USA, 2006; pp. 19-36.

[8] M. Shtern and V. Tzerpos, "Clustering methodologies for software engineering," Advances in Software Engineering, January 2012, Vol. 2012, Article 1, 18 pages.

[9] G. Czibula and G. Serban, "Hierarchical Clustering for Software Restructuring," Babes Bolyai University, Romania, 2007;

[10] W. E. Donathand A. J. Hoffman, "Lower bounds for the partitioning of graphs," IBM Journal of Research and Development. 1973, Vol.17, pp. 420-425.

[11] M. A. Fiedler, "property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," Czechoslovak Mathematical Journal. 1975, Vol. 25, pp. 619-633.

[12] M. Fiedler. "Algebraic connectivity of graphs," Czechoslovak Mathematical Journal, 1973, Vol. 23, pp. 298-305.

[13] B. Hendrickson and R. Leland, "An improved spectral graph partitioning algorithm for mapping parallel computations," SIAM Journal on Scientific Computing, 1995, Vol. 16, pp.452-469.

[14] A. Pothen, H. D. Simon and K. P. P. Liu, "Partitioning sparse matrices with eigenvectors of graphs source," SIAM Journal on Matrix Analysis and Applications, 1990, pp. 11, 1-30.

[15] S. Barnard and H. Simon, "A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems," Concurrency: Practice and Experience, 1994, Vol 6, pp.101-117.

[16] S. V. Dongen. Graph clustering by flow simulation. PhD thesis, University of Utrecht, Netherland.

[17] R. Reams, "Hadamard inverses, square roots and products of almost semi definite matrices," Linear Algebra and its Applications, 1 February 1999, Vol. 288, pp. 35-43

[18] Han, M. Kamber (2006) 'Data Preprocessing', Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, an imprint of Elsevier, pp.105-140.

[19] A. S. Muttipati and P. Padmaja. "Construction of Software Model Graph and Analysing Object-Oriented Program (C#) Using Abstract Syntax Tree Method," International Journal of Computer Science and Information Technologies, 2015, Vol. 6, pp. 3288-3293.