Discovering the Most Dominant Nodes in Frequent Subgraphs

Farah Chanchary
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
Email: farah.chanchary@carleton.ca

Herna Viktor
School of Elec Eng. and Comp Sc.
University of Ottawa
Ottawa, Canada K1N 6N5
Email: hviktor@uottawa.ca

Anil Maheshwari
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
Email: anil@scs.carleton.ca

***Abstract*—Recently, there is a growing trend to utilize data mining algorithms to explore datasets being modeled using graphs. In most cases, these graphs evolve over time, thus exhibiting more complex patterns and relationships among nodes. In particular, social networks are believed to manifest the preferential attachment property which assumes that new graph nodes have a higher probability of forming links with high-degree nodes. Often, these high-degree nodes have the tendency to become the articulation points in frequent subgraphs (also known as the most dominant nodes). Thus, their identification is important, because their disappearance may have greater influence on their peer nodes. Also, exploring their properties is essential when aiming to predict future frequent patterns. In this paper, we introduce a binary classification model *DetectMDN* to correctly classify the most dominant nodes in frequently occurring subgraphs. A set of experimental results confirms the feasibility and accuracy of our approach.**

Keywords–Frequent subgraphs; Graph mining; Most dominant nodes; Time evolving graph.

I. Introduction

In recent years, there has been considerable interest in graph structures arising in technological, sociological, and scientific settings. These domains include computer networks (routers or autonomous systems connected together); networks of users exchanging e-mails or instant messages; citation networks and hyperlink networks and social networks [25]. Frequent pattern mining provides a way to extract significant and interesting patterns from large datasets that otherwise commonly remain unexplored. This idea was first proposed by Agrawal and Srikant [1] in their work on proposing faster association rules mining. At present, frequent pattern mining is used in numerous scientific, business and legal application domains where the datasets are generally large, multidimensional and dynamic, and traditional approaches of exploratory data analysis yield limited success [14].

Other than association rule mining, frequent pattern mining is also used in many standard knowledge discovery tasks such as classification and clustering to achieve better outcomes from these patterns. At the same time the applications have expanded extensively in domains where objects of various datasets are modeled using graphs [21] [22], trees [19], sequences [16] or time evolving networks [7] [24]. Following the trend, research has been carried out on graph and tree pattern mining to discover complex associations that are frequent according to some predefined concept.

Interesting frequent patterns can further be used as a source of features in a supervised learning task when the database events are labeled. In many networks, e.g., communication, social networks, citation and co-authorship networks, this idea can play a significant role in solving problems like hidden group identification and link prediction. In cases when similar patterns of groups or links occur very frequently within large networks, they can provide important information for predicting future patterns. This leads to the problem of efficiently identifying most dominant nodes (MDNs) in these frequent sub-groups. It follows that MDNs tend to dominate the interests and activities of their peer nodes. Thus, knowing their attachment patterns is valuable to many entities, including marketers, employers, credit rating agencies, insurers, spammers, phishers, police, and intelligence agencies [8].

The main contributions of this study are three-fold. Our paper describes and presents the performance of *DetectMDN* algorithm designed to find, and subsequently correctly label MDNs in hidden frequent subgraphs of any large network. We further analyze the impact of different types of networks on the prediction results. Thirdly, we identify the most relevant attributes to accurately define frequent subgraphs.

The paper is organized as follows. Section 2 describes the problem domain and highlights related work. This is followed, in Section 3, with a description of our *DetectMDN* algorithms. Section 4 details our experimental setup and evaluation, while Section 5 concludes the paper.

II. Description of the Problem Domain and Related Work

There are generally two distinct settings for subgraph mining. The first one is the *graph-transactional* setting, where a set of graphs $\{G_1, ..., G_n\}$ and a threshold t are given, and our goal is to find patterns that occur in at least t graphs in the set. The second graph mining setting is called the *single network* setting, where a single graph G and a threshold t are given, and we aim to find patterns that have a support of at least t in G according to some appropriate support measure. We use both settings as inputs to evaluate our classification model.

Formally, the problem definitions are as follows:

1) In a graph-transactional setting, a graph dataset $D_1 = \{G_1, ..., G_n\}$ and a threshold t are given.

2) In the single network setting, a dataset D_2 consisting of a large graph G and a threshold t are given.

In this setting, our task is thus to correctly identify all MDNs in D_1 and D_2.

All graph datasets used in this study are undirected. We revisit relevant definitions below.

Definition 1: A graph $G = (V, E)$ has a set of nodes denoted by V and the edge set denoted by E. A graph G' is a subgraph of another graph G if there exists a subgraph isomorphism from G' to G, denoted by $G' \subseteq G$.

Definition 2: A subgraph is frequent if its support (occurrence frequency) in a given dataset is no less than a minimum support threshold t, where $t > 0$.

Hence, the frequent subgraph is a relative concept, whether or not a subgraph is frequent depends on the value of t [28].

Definition 3: A subgraph isomorphism is an injective function $f : V(G) \to V(G')$, such that, (1) $\forall u \in V(G)$, $l(u) = l'(f(u))$, and (2) $\forall (u, v) \in E(G), (f(u), f(v)) \in E(G')$ and $l(u, v) = l'(f(u), f(v))$ where l and l' are the label function of G and G', respectively.

These labels represent some properties of the nodes (or edges) from the same domain, e.g., location or gender, and for simplicity it is assumed that they do not change with time. We also assume that labels do not represent any combinations of properties (i.e., both location and gender). Each node (or edge) of the graph is not required to have a unique label and the same label can be assigned to many nodes (or edges) in the same graph. Here, f is called an embedding of G in G'.

Figure 1 presents an example of frequency counts of subgraphs X and Y in a source graph G. Subgraph X on top has a frequency count of 2 and the subgraph Y on bottom has frequency counts of 4 in the source graph on the left.

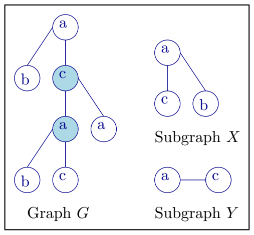

Figure 1. Frequency counts of subgraphs.

Definition 4: A most dominant node $u \in V$ in an undirected graph G is an articulation point having maximum degree in a frequent subgraph of G.

Definition 5: A node $x \in V$ in an undirected graph G is an articulation point if removing x disconnects the graph into two or more connected components. In other words, node $x \in V$ is an articulation point if there exists distinct nodes v and w, such that every path between v and w goes through x.

Figure 1 has three articulation points but only two of these are MDNs (shaded nodes). Note that more than one articulation point may have the same maximum degree, and these are all considered as MDNs.

Definition 6: A time evolving graph is a conceptual representation of a series of undirected graphs $G_0, \cdots, G_T$, so that $G_t = (V_t, E_t)$ represents the graph at time t over time 0 to T. Thus, $\{G_0, G_1, \cdots, G_T\}$ is combined as one undirected graph $G = (V, E)$ with $V = \cup_{t=0}^{T} V_t = V_T$ and $E = \cup_{t=0}^{T} E_t = E_T$. To each edge $e = (u, v)$ a time-stamp $t(e) = \min_j\{E_j | e \in E_j\}$ is assigned. So, formally a time evolving graph is defined as $G = (V, E, t, \lambda)$ with t assigning time-stamps to E and a labeling function $\lambda : V \cup E \to \Sigma$, assigning labels to nodes and edges from an alphabet Σ [24].

The edge labels of the graph shown in Figure 2 represent collaboration time between pairs of authors. Label 0 represents the starting time (e.g., month, day or year) and it increases by 1 after every time unit.

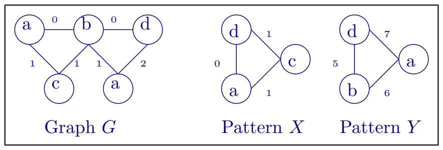

Figure 2. Time evolving graph.

Time evolving graphs support relative-time pattern matching.

Definition 7: Let $G = (V, E, t, \lambda)$ be a time evolving graph and $P = (V_P, E_P, t_P, \lambda_P)$ is a pattern subgraph. Let us assume that P is connected. P occurs in G at relative time if there exists a $\Delta \in R$ and a function $\psi : V_P \to V$ mapping the nodes of P to the nodes in G such that, $\forall u, v \in V_P$:

1) $(u, v) \in E_P$ implies $(\psi(u), \psi(v)) \in E$
2) $(u, v) \in E_P$ implies $t((\psi(u), \psi(v))) = t(u, v) + \Delta$
3) $\lambda_P(u, v) = \lambda(\psi(v))\lambda_P(u, v) = \lambda((\psi(u), \psi(v)))$

Figure 2 shows an example of how relative-time pattern matches subgraphs with the source graph. Here, pattern X is not a valid subgraph of G since node labels do not match properly. On the other hand, pattern Y is valid according to property 2 of Definition 7.

Definition 8: Minimum Image Based Support is based on the number of unique nodes in the graph $G = (V, E)$ that a node of the pattern $P = (V', E')$ is mapped to, and defined as $\delta(P, G) = min_{v \in V'} |\{\psi_i(v) : \psi_i \text{ is an occurance of } P \text{ in } G\}|$.

Figure 3 shows the host graph (a) and a pattern (b). According to minimum image based support measure the frequency of the given pattern in the host graph will be 1, though in general the pattern can be matched with multiple embeddings of the host graph.

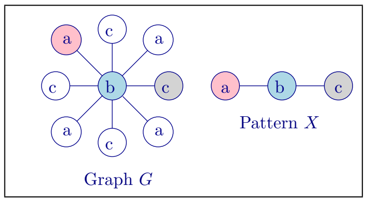

Figure 3. A graph with possible different occurrences of a pattern.

A. Related Work

A number of research has been carried out on the separate themes of the effect of node removal from large graphs and discovering influential or important nodes from large graphs based on different measurement and characteristics of graph elements, such as a model based on semi-local centrality measure [5], another model based on bio-inspired centrality measure [6], stochastic diffusion models [20], and the information transfer probability between any pair of nodes and the k-medoid clustering algorithm [33]. In [3], Albert et al. showed that a class of inhomogenously wired networks (such as the World Wide Web, social networks and communication networks) are extremely vulnerable to the selection and removal of a few nodes that play a vital role in maintaining the network connectivity. Shetty et al. [27] worked on an entropy model to identify the most interesting nodes in a graph. Here, the concept of importance depends on the amount of commands/messages forwarded through the network. A different measure using principal component centrality has been used in [15] to identify social hubs, nodes at the center of influential neighborhoods. Further, the problem of identifying influential spreaders in complex networks has also been studied in [34]. As far as we are aware, this paper presents the first work in which a classifier is used to find the MDN in frequent subgraphs.

III. DetectMDN Algorithm

This section describes main components of our DetectMDN algorithm.

Step 1. Extract frequent patterns: From input graph datasets $D_1 = \{G_1, ..., G_n\}$ and D_2, all frequent subgraphs are extracted where frequency occurs more than a given threshold t. By applying a graph mining algorithm (gSpan [32]) on D_1, we construct a set of frequent subgraphs $D'_1 = \{F_1, ...F_n\}$. Another graph mining algorithm (GERM [24]) is applied on D_2 to obtain another set of frequent subgraphs $D'_2 = \{F_1, ...F_m\}$.

We describe these algorithms here briefly. The *gSpan* algorithm uses a pattern growth approach to discover frequent subgraphs from large graphs, as depicted in Figure 4 and 5. The gSpan method builds a new lexicographic order among graphs, and maps each graph to a unique minimum depth first search (DFS) code as its canonical label. Based on this lexicographic order, gSpan adopts the DFS strategy to mine frequent connected subgraphs efficiently. This algorithm uses the DFS lexicographic order and rely on the minimum DFS concept, which forms a novel canonical labeling system to support the DFS search [32].

```
Data: Graph set $\mathcal{GS}$
Result: Set of frequent subgraphs $\mathcal{S}$
sort labels in $\mathcal{GS}$ by their frequency;
remove infrequent nodes and edges and relabel
remaining nodes and edges;
$S' \leftarrow$ all frequent 1-edge graphs in $\mathcal{GS}$;
sort $S'$ in DFS lexicographic order;
$S \leftarrow S'$;
for each edge $e \in S'$ do
    initialize $s$ with $e$, set $s.\mathcal{GS}$ by graphs which
    contains $e$;
    SubgraphMining($\mathcal{GS}, \mathcal{S}$,s);
    $\mathcal{GS} \leftarrow \mathcal{GS} - e$;
    if $|\mathcal{GS}| < minSupp$ then
        break;
    end
end
```

Figure 4. GraphSet_Projection Algorithm

On the other hand, *GERM* mines patterns from a single large graph. The GERM algorithm is adapted from gSpan where the support calculation is replaced by the minimum image based support (see Definition 8). GERM has made another modification in gSpan's use of minimum DFS code by modifying the canonical form used in DFS code. This is done so that the first edge in the canonical form is always the one with the lowest time-stamp, as compared to gSpan where the highest label is used as a starting node [22].

```
Data: Graph set $\mathcal{GS}$, DFS code $s$
Result: Set of frequent subgraphs $\mathcal{S}$
if $s \neq min(s)$ then
    return;
end
$S \leftarrow S \cup s$;
enumerate $s$ in each graph in $D$ and count its children;
for all $c, c$ is $s'$ child do
    if $support(c) \geq minSupp$ then
        $s \leftarrow c$;
        $SubgraphMining(\mathcal{D}, \mathcal{S}, s)$;
    end
end
```

Figure 5. SubgraphMining Algorithm

Step 2. Apply LabelMDN algorithm: The algorithm works as follows (see Figure 7).

For each frequent pattern F_i of every datasets D'_1 and D'_2, it identifies the set of articulation points A. If $|A| = p > 1$, we identify the total number of nodes $V(F_i)$ and edges $E(F_i)$, and number of connected components $BC(F_i)$. We also calculate the degree $deg(a_j)$ and associated time stamps $ts(a_j)$ of each articulation point $a_j \in A$, where $1 \leq j \leq p$. An articulation point can be associated with multiple time stamps as it evolves with time.

To compute the articulation points, we implement the standard linear time DFS approach proposed by Hopcroft and Tarjan [18]. All articulation points satisfy the following conditions: Any node x is an articulation point if and only if either (a) x is the root of the DFS tree and x has more than one child, or (b) x is not the root and for some child w of x there is no back-edge between any descendent of w (including w) and a proper ancestor of x. (For further reading, we refer the interested reader to [4].)

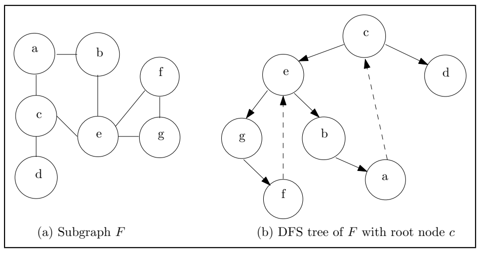

Figure 6. Labelling articulation points using DFS tree.

Figure 6 gives an example of a given subgraph F in (a) and its DFS tree starting at c in (b). Backedges are shown with dotted lines. It is clear that F has two articulation points c and e, but only e will be labeled as MDN since it has the maximum degree in F.

The next step is to correctly label each a_j whenever it is a MDN in F_i. This is done by verifying whether $deg(a_j)$ of each articulation point a_j is maximum within it's corresponding F_i.

Thus, each a_i is labeled with a class label C_1 if it is a MDN, with C_0 otherwise. Now, we have a training dataset T having records for each a_j with the expectation to train a classifier for correctly predicting the labels of unclassified test datasets. *Step 3. Classify MDNs:* A number of classification algorithms

```
Data: Set of frequent subgraphs $D'_1$, $D'_2$
Result: Training dataset $T$
for all frequent subgraphs $F_i \in D'_1$ (or $D'_2$) do
    find set of articulation points $A \in F_i$. [use the standard DFS based algorithm];
    if $|A| \geq 1$ then
        find $V(F_i)$, $E(F_i)$, and $BC(F_i)$;
        for each $a_j \in A$ from $j = 1$ to $p$ do
            find $deg(a_j)$ and $ts(a_j)$;
            if $deg(a_j) = \max\{deg(u) : u \in V(F_i)\}$ then
                label $a_j$ with class $C_1$. [$a_j$ is a MDN];
            else
                label $a_j$ with class $C_0$. [$a_j$ is not a MDN];
            end
            store information of each $a_j$ as a record in $T$;
        end
    end
end
```

Figure 7. LabelMDN Algorithm

was used from the WEKA environment [31]. The baseline model is constructed using ZeroR, where the classifier assigns all items to the largest class. We also utilized a number of other classifiers, including the Naive Bayes probabilistic approach, Support Vector Machines (SMO), a rule based method (JRip) and a decision tree based classifier (J48). We also built models using the Boosting and Bagging meta-learning techniques, both using J48 decision trees as base learner.

IV. Experimental Setup and Evaluation

This section describes our experimental setup and evaluation. We have used the implementations of gSpan and GERM that are available in [13] and [24] respectively. We made use of the previously mentioned WEKA classifiers, as noted earlier. All other algorithms have been implemented in Java programming language.

A. Datasets

Five datasets have been used for evaluating *DetectMDN* algorithm. Brief descriptions of these datasets are given below, and their summary information is shown in Table I.

(a) Co-authorship network data, DBLP [11]: Three sample datasets from the same DBLP network span over three periods, namely 1992-2002, 2003-2005 and 2005-2007. In this dataset, authors are represented by nodes with a connecting edge if they are co-authors. The assigned time-stamp on an edge represents the year of the first co-authorship. The three different samples each contains the edges created in the corresponding years. We also aggregate all datasets into one large DBLP dataset with information from the year 1992 to 2007. This is done for analyzing both long and short term trends.

(b) Social network dataset, Facebook [12]: This network describes friendship relations between users of Facebook. It was collected in April of 2009 through data scraping from Facebook. Each node represents a user and an edge represents friendship between two users. The graph is undirected and unweighted thus all nodes are labeled with the same default value.

(c) Citation network [10]: The format for the Citation dataset is similar to the Facebook dataset. It contains a sorted 2 column vector where the left column is the *arxiv id* of the paper cited from and the right column is the *arxiv id* of the paper being cited.

TABLE I. Dataset Properties.

Dataset	*Date*	*Node*	*Edge*	*Avg Degree*	*Time Stamp*
dblp92-02	92-02	129073	277081	2.15	0-10
dblp02-05	02-05	109044	233961	2.15	0-3
dblp05-07	05-07	135116	290363	2.15	0-2
facebook	2009	4039	88234	21.85	NA
citation	92-03	27718	352806	12.73	NA

B. Experimental Results

This section generalizes the results we obtained against the above-mentioned datasets. Firstly, the GERM algorithm has been applied to all the DBLP datasets to extract frequent subgraphs. These files are compatible with GERM and need no further modification. In the DBLP datasets the timestamps vary from 0 to 10. To normalize this range, each timestamp has been categorized as *initial*, *middle* and *final* and a three-digit bit string is generated which later is converted into a numeric value. Note that, since the Facebook and Citation datasets do not have timestamps, the *ts* attribute in Algorithm 7 is not relevant for these datasets. Subsequently, the original gSpan algorithm has been used for these two datasets. Table II gives brief explanations of the set of attributes generated by LabelMDN from all frequent patterns. Table III gives a summary of the results found in this level. The numbers under *Nodes*, *Edges*, *Degrees* and *BC* columns come with the format min-max-average. The *ts* column shows only min-max numbers. Although there is no significant differences in the average numbers of nodes and edges in frequent subgraphs of all five datasets, articulation points in Facebook and Citation datasets have higher degrees than that in the DBLP datasets.

TABLE II. Attributes extracted from given datasets.

Attributes	*Description*	*Type*
$V(F_i)$	Total number of nodes in frequent subgraph F_i	Numeric
$E(F_i)$	Total number of edges in frequent subgraph F_i	Numeric
$BC(F_i)$	Total number of biconnected components in F_i	Numeric
$deg(a_j)$	Total degrees of articulation point a_j	Numeric
$ts(a_j)$	Time stamps associated with a_j	Numeric
C_k	Class of a_j, where $k \in \{0, 1\}$	Binary

TABLE III. Summary of extracted attributes.

Datasets	*Inst*	*Nodes*	*Edges*	*Degrees*	*BC*	*ts*
dblp92-02	2471	3-10-5.8	2-9-4.9	2-5-2.4	2-9-4.7	1-5
dblp03-05	1971	3-11-5.7	2-10-4.9	2-6-2.5	2-10 4.6	1-7
dblp05-07	2923	3-12-5.7	2-11-4.9	2-7-2.5	2-11-4.6	1-7
Facebook	65	3-14-6.3	2-13-5.5	2-13-4.2	2-13-5.2	—
Citation	80	3-13-6.0	2-12-5.1	2-12-3.8	2-12-4.9	—

C. Research Questions

In this study, experiments are conducted to answer the following research questions:
Q1. Do our *DetectMDN* algorithm result in accurate models of the studied networks? (See Section C.1)
Q2. Do different types of networks (co-authorship and social) influence the prediction results? (See Section C.2)
Q3. What type of frequent pattern attributes are more effective than others? (See Section C.3)

C1. Performance of DetectMDN Algorithm

Table IV shows the accuracies achieved by each of the classifiers against all the datasets. In this table the names *DBLP1, DBLP2, DBLP3, FB* and *Cita* are used to represent *dblp92-02, dblp03-05, dblp05-07, Facebook* and *Citation* datasets respectively. All WEKA classification algorithms are used with 10-fold cross validation for each of the datasets. Compared with the baseline all other classifiers performed significantly better, and for all datasets, the best performances are shown by JRip and AdaBoost (with J48) both are ranked as 1. It should be noted that the differences among these top ranked classifiers' performances are not significant. Our algorithm successfully classifies MDNs in both full-term and short-term time-evolving DBLP datasets, while the classifier performs equally well for the static graph datasets, namely Facebook and Citation. These results show that our classifiers are able to encompass both the duration and the variation of the relationship patterns among all MDNs.

TABLE IV. ACCURACY OF CLASSIFICATION MODELS (%).

Classifiers	*DBLP1*	*DBLP2*	*DBLP3*	*DBLP*	*FB*	*Cita*	*Rank*
Base	67.5	65.7	66.8	66.7	81.8	82.5	4
JRip	74.3	74.4	74.2	73.8	90.8	85.0	1
NaiveBayes	71.6	72.0	71.8	72.0	80.0	78.8	4
J48	74.1	74.4	74.2	73.6	88.0	86.3	2
SMO	73.3	74.0	74.3	74.0	81.5	82.5	3
Boosting	73.5	74.5	74.3	73.6	90.8	85.0	1
Bagging	73.7	74.4	74.2	73.8	87.7	81.3	2

C2. Influence of Network Types on Prediction Results

Since a classifier's performance does not only depend on the percentage of accuracy value, we present another set of analysis in Table V. We consider two classifiers, namely Boosting (AdaBoostM1) with a J48 base learner, as well as J48 on its own, for this analysis. These two classifiers were chosen based on their high rankings during our earlier experiments.

ROC analysis and AUC: For all datasets, the high percentage of Area Under Curve (AUC) value demonstrates the efficiency of our model since it depicts the relative trade-offs between true positives and false positives across a range of thresholds of a classification model. Figure 8 shows the comparison between the ROC graphs of the AdaBoostM1 (J48) and J48 algorithms for the Citation, Facebook and DBLP datasets.

F-measure represents a harmonic mean between recall and precision.

$$F - measure = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

A lower recall value (69%) for DBLP results a comparatively lower F-measure (75%). For the static datasets, the accuracy rates are more than 90% for Facebook dataset and 85% for Citation dataset. Figure 9 plots the Precision-Recall (PR) graph demonstrating performance of the classifier for all datasets. From these results it can be said that static networks are more predictable than time evolving graphs.

TABLE V. AUC AND F-MEASURE VALUES USING ADABOOST WITH J48.

Dataset	*AUC*	*Precision*	*Recall*	*F-measure*
DBLP	0.82	0.78	0.69	0.75
Facebook	0.96	0.90	0.91	0.91
Citation	0.93	0.85	0.85	0.85

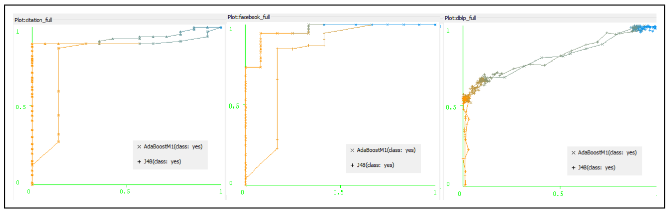

Figure 8. Comparisons between ROC Graphs of AdaBoostM1 and J48 for (a) Citation, (b) Facebook and (c) DBLP. The x-axis represents the false positive (FP) rates and y-axis represents the true posotive (TP) rates.

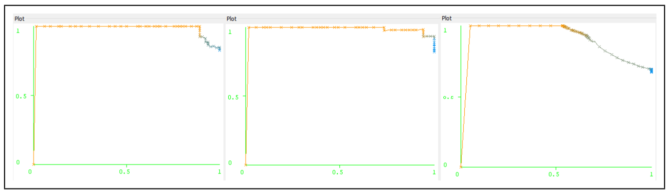

Figure 9. Precision and Recall graph for (a) Citation, (b) Facebook and (c) DBLP. Here, x-axis represents Recall and y-axis represents Precision.

C3. Effective Attributes for Classification

The aim of this section is to evaluate the contribution of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. We employed the *CfsSubsetEval* function for this task, which has been successfully applied previously in many studies including [9] [17] [26] to improve the accuracy. For all datasets, the *degrees* attribute has been identified as the most important attribute to aid the classifier, while DBLP dataset utilizes a second attribute, namely *nodes*. Both the Facebook and Citation datasets have comparatively higher average degrees than the DBLP dataset (see Table I). Recall that the first two datasets represent a static view of the network where the communication with peer nodes are not labeled with time. Therefore, the total number of peer nodes (i.e., degrees) associated with each articulation point mostly governs the classification rules. In the case of the DBLP datasets, our classifier performs the labeling task based on the total number of authors (*nodes*) and consider when they are connected with peer nodes (*degrees*). It should be noted that dependency on different attribute sets do not degrade the overall performance of our classifier.

D. Discussions

In this study, we are interested to correctly classify all most dominant nodes in frequent subgraphs of large networks. Based on the results reported above it can be concluded that our model can successfully classify both static and time-evolving graphs with reasonable accuracy. However, the large number of frequent subgraphs that are often present in network settings constitutes some challenges [28]. The first one is *redundancy*, in that most frequent subgraphs only differ slightly in structure and repeat in many subgraphs. The second is the *statistical significance* of subgraphs. Since both frequent and infrequent subgraphs may be uniformly distributed over all classes, only frequent subgraphs whose presence is statistically significantly correlated with class membership are promising contributors for classification.

In our research, MDNs are defined irrespective of their direction of communications with peer nodes. Although it has little impact in co-authorship networks, we may expect a conceptual change in social and communication networks where a MDN either communicates with (both-way) or only broadcasts to (one-way) their peer nodes in a frequent subgraph. For example, a fan page of Facebook may have many responses from a group of members for each post. On the other hand, an online marketing page may post news to many of its clients, but typically does not receive responses equally.

V. Conclusion and Future Work

We presented an algorithm to classify the most dominant nodes in frequent subgraphs of large networks. We considered both static and time-evolving graphs. We evaluated the performance of our algorithm with percentage of accuracy, precision and recall. We also identified the attributes (i.e., *degrees* in general, and a second attribute *nodes* in DBLP dataset) that are the most effective for successful classification. Our experimental results showed that our method achieved good performance in terms of accuracy and other statistical measurements. Our future work will center on finding scalable solutions to effectively deal with numerous frequent subgraphs that are similar in structure and scope. We are also interested in studying the effects of changes in properties, in order to extend our work to deal with concept drift in a graph setting.

References

[1] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases." *In Proc. of ACM SIGKDD Conference on KDDM*, pp. 12-19, 1994.

[2] A. L. Barabasi and R. Albert, "Emergence of scaling in random networks." *Science* 286, no. 5439, pp. 509-512, 1999.

[3] R. Albert, H. Jeong, and A. L. Barabasi, "Error and attack tolerance of complex networks." *Nature*, 406(6794), pp. 509-512, 2000.

[4] Biconnected Component, (2016, Mar.) [Online]. Available: https://en.wikipedia.org/wiki/Biconnected_component

[5] D. Chen, L. Lu, M. S. Shang, Y. C. Zhang, and T. Zhou, "Identifying influential nodes in complex networks," *Physica a: Statistical mechanics and its applications*, 391(4), 2012.

[6] C. Gao, X. Lan, X. Zhang, and Y. Deng, "A bio-inspired methodology of identifying influential nodes in complex networks," *PloS one*, 8(6), 2013.

[7] B. Bringmann and S. Nijssen. "What is frequent in a single graph?." *In Advances in Knowledge Discovery and Data Mining*, Springer Berlin Heidelberg, pp. 858-863, 2008.

[8] J. Bonneau, J. Anderson, R. Anderson, and F. Stajano. "Eight friends are enough: social graph approximation via public listings." *In Proc. of the 2nd ACM EuroSys Workshop on Social Network Systems*, ACM, pp. 13-18, 2009.

[9] P. Chanda, Y. R. Cho, A. Zhang, and M. Ramanathan, "Mining of attribute interactions using information theoretic metrics," *IEEE International Conference on Data Mining Workshops*, 2009.

[10] Citation network, (2016, Jan.) [Online]. Available: http://www.cs.cornell.edu/projects/kddcup/datasets.html

[11] Co-authorship Network, (2016, Mar.) [Online]. Available: http://www-kdd.isti.cnr.it/GERM/

[12] Facebook Dataset, (2016, Mar.) [Online]. Available: http://snap.stanford.edu/data/egonets-Facebook.html

[13] gSpan: Frequent Graph Mining Package, (2016, Mar.) [Online]. Available: http://www.cs.ucsb.edu/~xyan/software/gSpan.htm

[14] M. A. Hasan, "Mining Interesting Subgraphs by Output Space Sampling," PhD Thesis, Rensselaer Polytechnic Institute, New York, 2009.

[15] M. U. Ilyas and H. Radha. "Identifying influential nodes in online social networks using principal component centrality." *IEEE International Conference on ICC*, IEEE, pp. 1-5, 2011.

[16] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu. "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth." *In Proc. on ICCCN*, IEEE, pp. 2-15, 2001.

[17] J. Hancke and D. Meurers, "Exploring CEFR classification for German based on rich linguistic modeling," *In Learner Corpus Research Conference*, 2013.

[18] J. Hopcroft and R. Tarjan. "Algorithm 447: efficient algorithms for graph manipulation." *Communications of the ACM* 16 (6), pp. 372-378, 1973.

[19] A. M. Kibriya and J. Ramon, "Nearly exact mining of frequent trees in large networks." *Machine Learning and Knowledge Discovery in Databases*, Springer Berlin Heidelberg, pp. 426-441, 2012.

[20] M. Kimura, S. Kazumi, N. Ryohei, and H. Motoda. "Extracting influential nodes on a social network for information diffusion," *Data Mining and Knowledge Discovery* 20, no. 1, pp. 70-97, 2010.

[21] M. Kuramochi and G. Karypis, "Frequent Subgraph Discovery," *In Proc. IEEE International Conference on Data Mining, ICDM*, pp. 313-320, 2001.

[22] M. Kuramochi and G. Karypis. "An efficient algorithm for discovering frequent subgraphs," *IEEE Transactions on Knowledge and Data Engineering*, 16.9, pp. 1038-1051, 2004.

[23] KDD cup 2003 Datasets, (2016, Mar.) [Online]. Available: http://www.cs.cornell.edu/projects/kddcup/datasets.html

[24] B. Michele, F. Bonchi, B. Bringmann, and A. Gionis. "Mining graph evolution rules," *In Machine learning and knowledge discovery in databases*, Springer Berlin Heidelberg, pp. 115-130, 2009.

[25] M. E. J. Newman. "The structure and function of complex networks," *SIAM Review* 45, no. 2, pp. 167-256, 2003.

[26] K. Selvakuberan, M. Indradevi, and R. Rajaram, "Combined feature selection and classification - A novel approach for categorization of web pages." *Journal of Information and Computing Science.* 3 (2), 2008.

[27] J. Shetty and J. Adibi, "Discovering important nodes through graph entropy the case of enron email database," *In Proc. of the 3rd international workshop on Link discovery*, ACM, pp. 74-81, 2005.

[28] M. Thoma, H. Cheng, A. Gretton, J. Han, H. Kriegel, A. Smola, L. Song, P. S. Yu, X. Yan, and K. M. Borgwardt, "Discriminative frequent subgraph mining with optimality guarantees," *Statistical Analysis and Data Mining* 3, no. 5, pp. 302-318, 2010.

[29] The Graph Evolution Rule Miner, (2016, Mar.) [Online]. Available: http://www-kdd.isti.cnr.it/GERM/

[30] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, Cambridge, ENG and New York: Cambridge University Press, Vol.8, 1994.

[31] Weka 3: Data Mining Software in Java, Machine Learning Group at the University of Waikato, (2016, Mar.) [Online]. Available: http://www.cs.waikato.ac.nz/~ml/weka/

[32] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," *IEEE International Conference on Data Mining*, pp. 721-724, 2003.

[33] X. Zhang, J. Zhu, Q. Wang, and H. Zhao, "Identifying influential nodes in complex networks with community structure," *Knowledge-Based Systems*, 42, 2013.

[34] Z. Zhao, X. Wang, W. Zhang, and Z. Zhu, "A Community-Based Approach to Identifying Influential Spreaders," *Entropy*, 17(4), 2015.