An Improved Pattern Mining Technique For Graph Pattern Analysis Using a Novel Behavior of Artificial Bee Colony Algorithm

Shriya Sahu, Meenu Chawla and Nilay Khare
Manit, Bhopal, India
E-mail: s.shriya88@gmail.com, chawlam@manit.ac.in, nilay.khare@rediffmail.com

Keywords: pattern mining, swarm intelligence, machine learning

Received: September 28, 2020

Rising data complexity and volume in the network has attracted researchers towards substructure analysis. Subgraph mining is an area that has gained remarkable attention in the last couple of years to offer an intelligent analysis of more massive graphs and complicated data structures. It has been observed that graph pattern mining faces issues regarding the matching ruleset and complex instruction set execution problem. This paper introduces modern-day intelligence architecture based on Swarm Intelligence that is cross-validated by supervised machine learning mechanisms. A new behavior incorporated with a new inter and intra hive behavior is incorporated in Swarm based Artificial Bee Colony. The proposed work model is evaluated over two different datasets with more than 4900 nodes in the graph. The proposed framework is evaluated using True Detection Rate, False Detection Rate, precision, and F-Measure, demonstrating an average improvement of 9.8%, 8.35%, 8.35% and 9.15% against existing GraMi work that represent an enhanced performance of the proposed pattern mining technique.

Povzetek: Uporabljene so raznovrstne metode umetne inteligence in strojnega učenja za iskanje vzorcev, tj. podgrafov v grafu.

1 Introduction

Frequent Subgraph Mining (FSM) plays a central role in solving complex problems for various applications such as text retrieval, computer vision, social networks, computational chemistry, and bioinformatics. Additionally, FSM also caters to the graphical problems in data mining tasks such as designing database, clustering, and classification of graphs. The main objective of the mining graphs is to compute the subgraphs whose appearances exceeds a certain threshold. Such a perspective is quite useful in understanding real-life applications. For example, protein-protein structures and their interactions easily modeled by labeled graphs. But it is a challenging task for uncertain graphs. Therefore, researchers focussed on efficient mining of frequent patterns on such graphs (Chen et al., 2018). Recently, there is a quite interest of the researchers to study the relationship between the entities and attributes in a social graph. Such a relation widely used in social media marketing likewise 90%, 14%, and 60% users said that customer's trust, advertisement, and Twitter respectively play a critical role in shopping. Nonetheless, the mining subgraph in social graphs is more related than rules in case of itemsets. Consequently, the researcher of bioinformatics may determine the substructures within protein interaction graphs and structures. Such graphs have nodes and edges which represent proteins and their interactions. In addition, these graphs are updated whenever there is a need to represent the interactions of new proteins. However, a critical task for the researcher is to forecast the working of a newly added protein without any experimentation. But it is possible only through frequent mining by interacting with the new proteins having identical interactions.

The problem of FSM is categorized into two phases, such as determining frequent patterns in either (a) graphical database having multiple inputs (Protein interaction or chemical compounds) or (b) large graph having single input (e.g., social media,) (Elseidy et al., 2014). The main task of FSM is to calculate all the subgraphs having support or frequency exceeds the minimum frequency threshold. In the case of multiple graphs, frequency is the count of pattern graphs (Ingalalli et al., 2018). But it is quite challenging to define the support notion in a single large graph. Thus, it is not enough to define the pattern that exists in a graph, whether it exists or not. Therefore, it is vital to determine all the isomorphisms (I) of *A*, which are distinct in nature from the pattern graph (G). Actually, 'I' is the exact match of *A* in the graph, which is used to pair the nodes, and edges with their respective labels (Cheng et al., 2014). For instance, if we talk about the collaboration graph (G) as depicted in Figure 1, subgraph (U_1) is having four isomorphisms. However, a typical approach to mine frequent subgraphs is using grow and store method which includes different phases such as (Gu et al., 2016; Yuan et al., 2012; Li et al., 2012) (a) Computation of all the nodes which exists at least user-defined threshold (£) and load their appearances (b) Frequently, extend the loaded

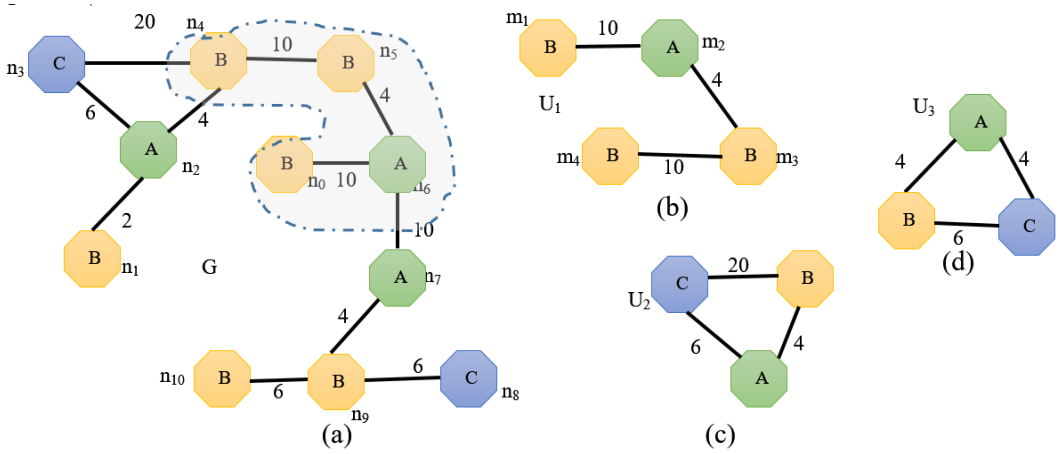

Figure 1: Directed Graphs with their isomorphism (a) Collaborative Graph (b) Subgraph 1 with four extensions (c) Subgraph 2 with three isomorphisms (d) Subgraph 3 with their appendices.

appearances to build large, frequent subgraphs and then assess their frequency (c) new frequent subgraphs appearances storage (d) Repeat the phase 2 until no more these subgraphs detected (Rehman et al., 2018; Abdelhamid et al., 2017).

A graph is the simplest form to represent the data by modeling relationships between the various objects. The interesting problem that comes in concern is pattern matching when handling graphical data. This matching sorted using the problem of subgraph isomorphism. For instance, there are two graphs, Y and Z, the role of subgraph isomorphism is to describe whether Z includes a subgraph which is isomorphic to Y, and it is an NP-hard problem. There are various algorithms proposed in the literature to solve this complex problem using genetic algorithm, MapReduce and Pregel (Bhuiyan & Hasan, 2015; Zhao et al., 2016; Choi et al., 2019). Moreover, generalized subgraph problem of the isomorphism sorted by developing some algorithms but these are limited to uncertain graphs which increases complexity, limited scalability and work with redundant data having supplementary data such as attributes or edge labels. Alternatively, researchers rely on metaheuristic algorithms such as Genetic algorithms to address the consequence of this problem. Most of the algorithms provide quality solutions with less time, but these are limited to the search capability in case of large space for the problem of subgraph isomorphism (Choi et al., 2019). Therefore, this paper solves this problem in frequent pattern mining using the Cuckoo search algorithm. The main advantage of using this algorithm is that it works efficiently within a large space in case of an NP-hard problem. The leveraged search capability detected the frequent patterns in a subgraph and evaluated the problem of subgraph isomorphism.

Preliminaries

A collaborative graph $G = (S, T, K)$ contains various nodes S, edges T with a labeling function K which assigns labels to S and T. A subgraph of G consists of Y and Z such as $Y \subseteq S, T$ and $Z \subseteq S, T$ if $S_Y and\ S_Z \subseteq S$ and $T_Y and\ T_Z \subseteq T$.

1.1 Subgraph Isomorphism (SI)

Definition 1: There are two graphs given such as G: Y= (S_Y, T_Y) and Z = (S_Z, T_Z), the SI is an injective function such as d: $S_Y \rightarrow S_Z$ such as $(m, n) \in T_Y$ in case of $(d(m), d(n)) \in T_R$ where $R = (S_R, T_R) \subseteq Z$. However, d is an induced SI in case if $(m, n) \notin T_Y$, then$(d(m), d(n)) \notin T_R$.

The basic difference between SI and induced SI is that edge absence in Y corresponds to the presence of an edge in Z must not present in case of induced SI. This mapping preserves the nodes and edges labels. For instance, subgraph Y has four isomorphism $(m_1 \underline{2}\ m_2 \underline{6} m_3 \underline{10} m_4)$ with respect to collaborative graph G, and $(n_1 \underline{2}\ n_2 \underline{6}\ n_3 \underline{20} n_4; n_4 \underline{10} n_5 \underline{4}\ n_6\ and\ n_0 \underline{10} n_6 \underline{10}\ n_7;\ n_8 \underline{6}\ n_9 \underline{10} n_{10})$. But, an intuitive way to determine the frequency of a subgraph in a graph is to count the number of isomorphism. In a given graph, the SI problem is the computation of subgraphs $Z \subseteq R$ such that $f: S_Y \rightarrow S_Z$ is an isomorphism from Y to R. This is rather a complex problem. Consequently, it is not an anti-monotone as the graphical representation shows that extension exceeds the subgraphs. For instance, in a given graph, node A appears 3 times while its extension (B) appears 4 times such that $A\ \underline{4}\ B$. The graph having such anti monotonic nature is of prime importance as it provides various methods without avoiding a situation. There are several anti-monotone metrics developed in literature (Talukder and Zaki, 2016; Elseidy et al., 2014).

Definition 2: There are two directed graphs such as G: Y= (S_Y, T_Y) and Z = (S_Z, T_Z) where $[S_Y \leq S_Z]$, the problem of SI represented by SI (Y, Z) is to determine an injective function $d: S_Y \rightarrow S_Z$ that reduces the value of fitness function (f). The optimal solution using the Cuckoo search having f=0 is the SI from Y to Z.$s_G(U) = \min\{t | t = |F(m)|\ for\ all\ m \in Y$

The fitness function (f) is defined as edges count, which match or may not match during the mapping. This

function is used to solve the real-world problems by constructing into SI problem and then solved it using the weights between the paths m and n. For instance,

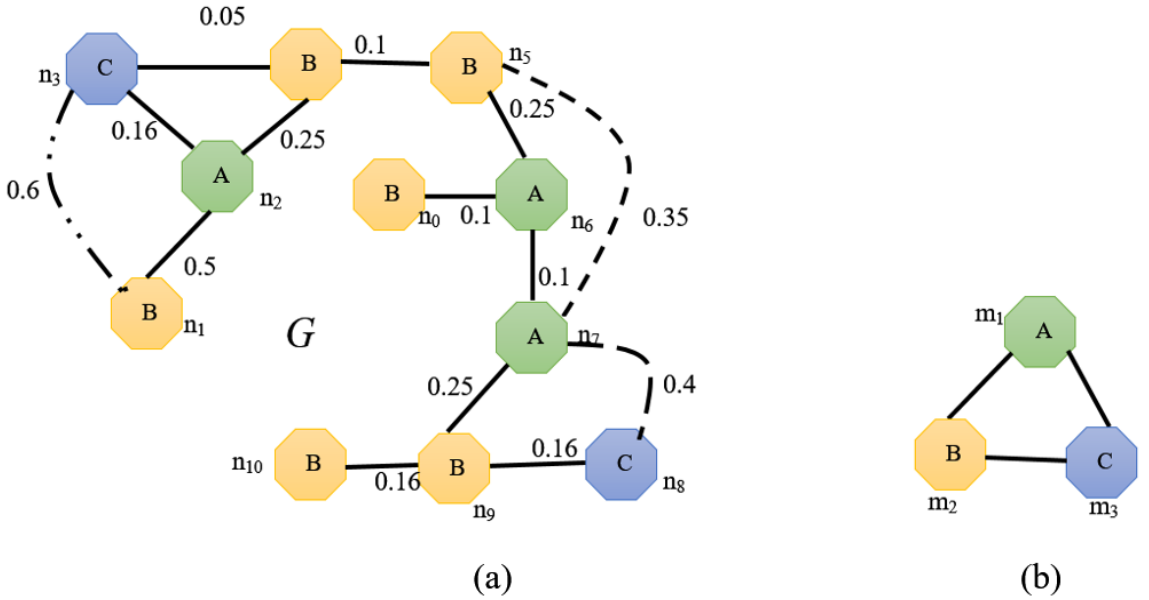

Figure 2: (a) Computation of distance for Graph G, as given in Figure 1. (b) Pattern graph.

metaheuristic approach. For instance, subgraph U_1 in Figure 1b and the graph G given in Figure 1(a) have $F(m_2) = \{n_2, n_6, n_7\}; F(m_1) = \{n_4, n_5, n_0\}; F(m_3) = \{n_4, n_5, n_9\}; F(m_4) = \{n_0, n_9, n_{10}\}$. Thus, $s_G(U_1) = 4$. In order to compare, the respective minimum support function is 2 $n_2 \underline{4} n_4 \underline{10}\, n_5$ *and* $n_6 \underline{4} n_5 \underline{10} n_0$as its isomorphism overlap and minimum support function regarded as unity. So, the main problem of FSM is given as follows:

Problem 1: In Figure 1, graph G is given with a minimum threshold (£), so the frequent subgraph mining problem is to compute all the subgraphs (U) in Graph G, such as $s_G(U) \geq (£)$.

Actual appearances which exceed the (£) does not compute in the given problem. This is quite impressive, and it is useful for many applications, but some prefer actual appearances such as graph indexing. Definition 1 relies on matching the labels of edges and nodes. For instance, subgraph U_2 has only a single isomorphism constructed by nodes n_2, n_3, n_4. However, research argues that developed matching is restrictive in nature and maintained by developing indirect relationships and differences of edges graphs and subgraphs. Such matching may also be possible for $n_7 \underline{4} n_9 \underline{6}\, n_8$, as seen in subgraph U_3. Rather, there is an indirect relation between A and C. This match often recognized as a pattern. For frequent mining patterns in this document, we use the definition from past research (Cheng et al., 2014).

Specifically, a distance matric has been employed, which computes the distance between two nodes, as given in the graph. In Figure 2 for graph G, a distance function that connects the m and n has been defined as $\Delta_h(m, n)$. The solid lines represent the relation using graph edges while dotted lines depict the transition. Let us consider that $\Delta_h(n_0, n_3)$ =4, then it is easy to use $\Delta_p(m, n)$ as the minimum sum of inversely proportional to the edge $\Delta_p(n_7, n_8) = \frac{1}{4} + \frac{1}{6} = 0.4$. Thus, a shorter distance belongs to robust collaboration.

Definition 3: In a pattern graph, $Q = (S_Q, T_Q, K_Q)$ of a graph G (S,T,K) if $S_Q \subseteq S, K_Q(m) = K(m) for\ all\ m \in S_Q\ and\ K_Q(edges) = \propto for\ all\ edges \in T_Q$.

In Figure 2, a pattern corresponds to a subgraph without any edge labels. However, Figure 2 (b) shows a pattern graph of a G.

Definition 4: Let us consider a pattern graph $Q = (S_Q, T_Q, K_Q)$of a graph G = (S, T, K), and Δ is a distance metric with a user − defined threshold (£). An injective function $(\varphi)$from pattern Q to G is $S_Q \rightarrow S\ only\ satisfied\ for\ K_Q(m) = K(\varphi(m))$ for all nodes $m \in S_Q$and $\Delta(\varphi(n), \varphi(m)) \leq £$.

The frequency and minimum support function of a pattern graph denoted by $\partial_G(Q) can\ be\ easily\ determined\ using\ definition\ 2$. Let us consider a threshold value £ = 0.3 then we get $\partial_G(Q) = 2$. Notify that there are only two constraints satisfy through this pattern graph such as $\Delta(\varphi(n), \varphi(m)) \leq £$.

In this paper, the GRAMI approach used in conjunction with the optimization technique to address the frequent mining problem in graphs. Additionally, GRAMI is a novel approach that solves the frequent mining problem by satisfying the constraints without affecting isomorphism in the graph.

This paper introduces a new algorithmic structure for the identification of the isomorphic patterns through the Cuckoo search algorithm. The rest of the paper is organized in the following manner. Section 3 represents the related work section, whereas section 4 represents the issue and solution as the proposed methodology of this paper.

2 Related work

In today's era, graph pattern mining is a frequent problem that comes in concern due to wide applications across various domains. In the literature, various approaches developed to address this problem, but still, there are enough gaps related to the isomorphism problem. For instance, Zhao had developed the Pregel based frequent subgraph mining approach to improve the scalability. Pregel is a computational model used to process the vertex graphs. A modern, distributed framework developed using this model to overcome the mining problem. The robust results obtained, but this approach still does not solve the constrained subgraph patterns on massive pattern graphs (Zhao et al., 2016). Aridhi and Nguifo had presented a study that summarized the existing data mining and graph processing techniques that could address the challenges faced by big graphs. Further, they provided a detailed classification of various graph processing designs along with vivid large-scale patterns or subgraph mining approaches (Aridhi and Nguifo, 2016). Moussaoui et al. had addressed the problem faced when subgraphs similarity could not be established. In this regard, researchers had proposed a flexible approach based on probabilistic graph mining to identify similar subgraphs. In this approach, probabilistic matching was implemented in comparison to the traditional exact similarity check. Experimentation against a real dataset of vivid domains had established that the proposed probabilistic model demonstrated better performance in terms of time processing and similar subgraph mining (Moussaoui et al., 2018). It has been observed that the structure and shape of the graph vary with respect to their applications. In this context, Jena et al. had introduced the SparkFSM approach that was proficient in dealing with isomorphism as well as directed and undirected graphs related to Spark or Scala technologies (Jena et al., 2018). Islam group had proposed WFSM-MaxPWS as an effective approach for subgraph mining based on weighted graphs. The mining approach proved to be very efficient in subgraph pruning. The approach was evaluated against different graph datasets representing normal and negative exponential weight distributions. Results had demonstrated that the runtime has significantly improved in comparison to the MaxW pattern mining approach (Islam et al., 2018). Iyer et al. had presented ASAP as an approximation-based subgraph and pattern mining technique. The authors also constructed an Error Latency Profile to specify the fluctuations observed for accuracy and current state in addition to approximating the graph patterns. Experimentation demonstrated that ASAP could successfully handle higher degree graphs comprising of billions of edges (Iyer et al., 2018). Researchers developed the metaheuristic-based algorithm to solve the isomorphism problem. The design issues have been considered to address the problem, which helps to decompose the consequences of a problem into the substructure. The optimal structure obtained using the hybrid genetic algorithm, which shows better results, but this approach has limited scalability and works in a concise search space (Choi et al., 2019). Preti et al. addressed the issue of pattern mining in large graphs representing multiple weight patterns. In the study, a scoring function-based pruning strategy was proposed that exemplified approximate as well as exact results to present subgraph mining (Preti et al., 2019). Detection strategies related to graphs were considered to be a very challenging task by Rao and Mishra. They had implemented pattern mining based on Edge Weight Detection (EdWePat) approach for identifying the subgraph patterns present in a weighted graph (Rao and Mishra, 2019). Li et al. had addressed the complex relationships existing in big graphs by introducing a fuzzy approach to traditional graph and pattern mining strategies. Authors had presented a multi-fuzzy based optimization using the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). The experimental evaluation demonstrated the effectiveness of the proposed strategy over the existing approaches (Li et al., 2019). Ray et al. had addressed the issue faced by subgraph mining that needs to be repeated frequently with respect to streaming larger graphs. In the process, they had developed a sampling design that could successfully mine out the subgraphs that represent the latest modification in the larger graph. Authors had involved 5 large graph datasets and a network motif mining algorithm to evaluate the proposed design. The results demonstrated that the proposed design could speedily identify the changing patterns (Ray et al. 2019). Priyadarshini and Rodda had proposed a Geometric Multi-Way Frequent Subgraph Mining (GMFSM) method. This method took advantage of the Frequent Subgraph Mining and filtration technique to shortlist the subgraphs from a single large database. The approach proved to be very effective and robust in achieving the required results and reduced the mining time from $1/3^{rd}$ to $1/2$ in comparison to the existing approaches (Priyadarshini and Rodda, 2020). Le and his group had postulated a Weighted Graph Mining (WeGraMi) algorithm as an effective approach for subgraph pruning. The design first calculated the weights of the pruned subgraphs, followed by applying search space analytics for subgraph pruning. The subgraph mining approach, based on the weighted threshold, had effectively addressed the issues concerning storage space and processing time (Le et al., 2020). Consequently, the probabilistic approach was investigated for frequent mining patterns on the uncertain graphs. An enumeration evaluation algorithm was proposed to address the semantic problem. Additionally, the computation sharing approach was used to obtain better performance.

The issue of mining and proposed solution

2.1 Dataset

The dataset is gathered from the following data sources.

a) http://data-mining.philippe-fournier-viger.com/subgraph-mining-datasets/ (dataset-1)
b) http://www.kaggle.com (dataset-2)

Both the dataset links have more than 5000 data elements and are open for download and processing.

The dataset-1 contains two standard subgraph mining data, which is provided for a small graph dataset. The file

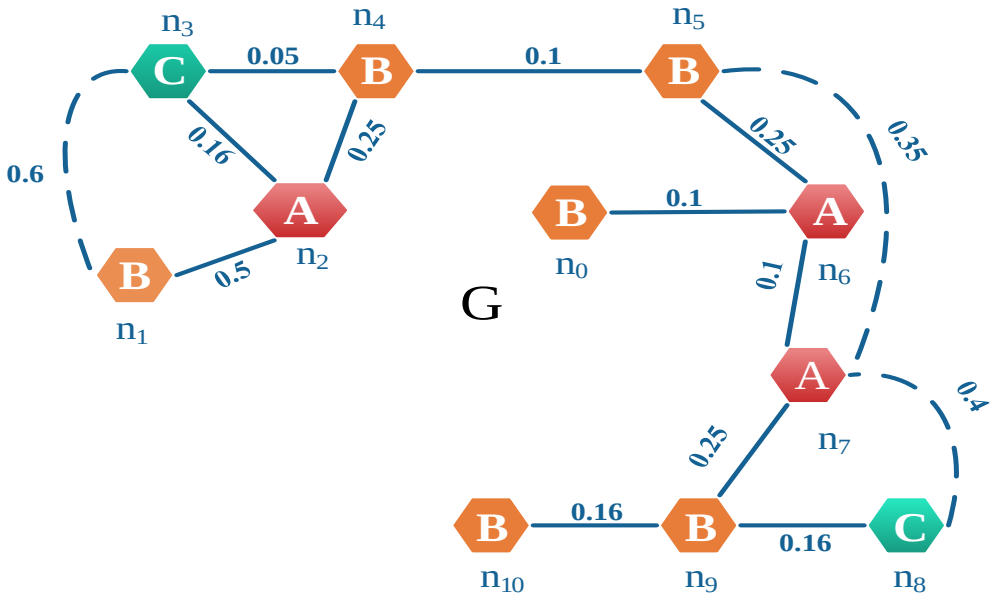

Figure 3: (a). Normal Pattern which has subsequent sections.

is available in the form of text, which composed of one or more graph. The graph is available in different format such as t≠ N, vML, ePQL.

t≠ N→ It represents the first line and is the N[th] graph in the file.

vML→ It represents the M[th] vertex of the recent graph with a label L

ePQL→ This attribute represents an edge with P[th] and Q[th] vertex for L number of labels.

Dataset-2 has been collected from Kaggle site. Comma-separated list is the simplest and supported file type available in Kaggle. Kaggle-loaded CSV's must have a header column with field names which can be easily read by human. The CSV file composed of two columns each contains metadata and description of data.

2.2 Issue and solution

Big graphs have always been an era of interest for different research world field experts. In order to understand the exact laying pattern of the big graphs, the normal mapping will result in a faulty rate of classification architecture. As the false placement of the pattern value can be done smartly and hence the standard mining architecture is not suitable enough for such kind of processing. This paper presents an improved behavior of the Artificial Bee Colony (ABC) algorithm to identify the pattern of the graphs. The general architecture of artificial bee colony has three kinds of bees as follows

a) The employed bee
b) The onlooker bee
c) The scout bee

The employed bee is the one who is responsible for the food collection, onlooker bee is for the monitoring purpose, and scout bee is searching for food sources randomly. This paper presents a new behavioral architecture of the artificial bee colony. At the initial phase, the entire graph is divided into 4 subsequent parts taking the initial point to be random, as shown in Figure 3(a) and (b).

As shown in Figure 3(b), the entire graph is divided into 4 different populations as Area 1, 2, 3, and 4. Now the bee colony algorithm will form 4 hives in each section, and the inter, as well as intra mining, will be formed. There is a semi queen for each population area, which determines the threshold of the mapped graph in each section, proceeded by the 20% selection rule. Apart from this inter clustering mechanism for bees, there is an intra mechanism as well. Pseudo Code 1 illustrates the working of ABC for the intracluster region.

PSEUDO CODE 1:

$Function\ Apply_{ABC}$

1. $Inputs: Sub_{Graph}\ Nodes, Output: Matched_{Pattern}$
2. $Foreach\ node\ in\ Sub_{Graph}$// For every node in Node List
3. $Initial\ Nectar_{Population} = Random_{Sample}$ // Initializing the Nectar
4. $Employed_{Bee} = node.Edge_{Weight}$ // The employed bee will be the edge weight of containing nectar
5. $Find\ In_{Degree}$; // Find inputs to the containing vertex
6. $Find\ Out_{Degree}$; // Find outs to the containing vertex
7. $Total_{Food_{Per_{Nectar}}} = In_{Degree}.Edge\ Weight + Out_{Degree}.Edge\ Weight$ // Total food in the hive will be equal to the edge weight of inward degree and the outward degree
8. $Store\ to\ Food_{Container}$ // Add the calculated value to the Food Container
9. End
10. $Foreach\ Bee_{Food}\ in\ Total_{Food_{Per_{Nectar}}}$ // Two different ranges are created

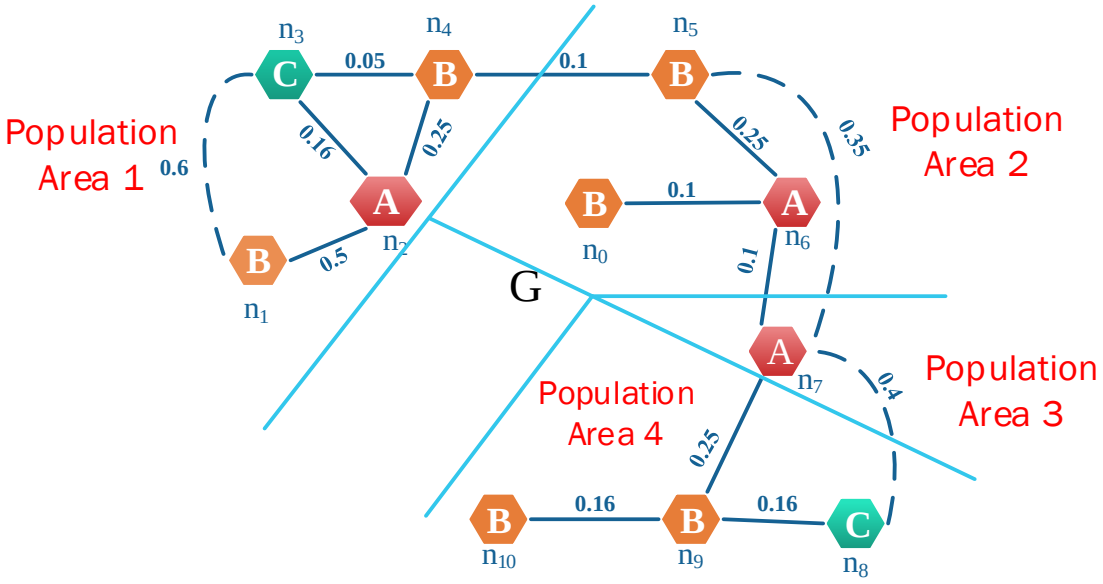

Figure 3: (b). Initially divided sections.

11. $Range1 = Bee_{Food} + Bee_{Food} * .20$ // The first is 20 % above the provided belt
12. $Range2 = Bee_{Food} - Bee_{Food} * .20$ // Second is 20 % below the provided belt
13. $Search_{Result} \leftarrow Range1 <= Data_{Value} <= Range2$ // Searching any other value in the same range
14. $Foreach\ SR\ in\ Search\_Result$
15. $Suspect_{Match}++$ // This could be a suspect similar graph pattern
16. End_{For}

The artificial bee colony creates a random population for the processing of the graph pattern. Each edge weight value will act as food to the nectar. The food calculation is done by summing up the edge weights of the in-degree and the out-degree of the nectar. The in-degree is increased by one if any node gets an edge from any other node in the graph. The out-degree is then incremented by one if the current node has an edge for any other node in the graph. Two range belts are created out of which the first proposed belt is 20% above and the second belt is 20% below the given belt. The search is done on the base of the calculated two new belts. The working is also represented by the flowchart, which is illustrated in Figure 4.

The found architectures could be a match of graph pattern, but it can't be termed as a final match. To find whether it is an exact match or not, the connecting edge value is passed to neural network. The ordinal measures of neural are defined in Table 1.

The pseudo-code for the architecture of neural network is given by Pseudo Code 2.

PSEUDO CODE 2

1. $Function\ Apply\ Neural\ Networks$
2. $Input: Suspect\ Nodes\ Output: Matched\ Graph\ Value$
3. $Set\ Training_{Value}\ to\ Empty$ // Initialize the Training Value to Empty, the matched architecture's edge weight will be passed as the training value
4. $Set\ Target_{Value}\ to\ Empty$ // The associated target value will be initialized to null
5. $Assign\ Training_{Value}\ associated\ with\ Suspect\ Nodes$
6. $For_{each}\ sp\ in\ Suspect_{List}$
7. $Training_{Value}.Append \leftarrow Set.Edge_{Value}$ // Assigning Edge Value
8. $Associated_{Target_{Value}} \leftarrow Set.sp.Id$
9. // Setting the target value as the edge value
10. $Start\ Training_{Architecture}$ // Starting the training architecture as per Ordinal Measures of Table 1
11. $If\ is\ satisfied\ Gradient\ Value$ // Check whether the gradient is satisfied or not
12. $Training_{Complete}$ // If the gradient is satisfied, the training is complete
13. End
14. $Store\ Trained\ Architecture\ as\ per\ Supervised\ Machine\ Learning$ // Applying Machine Learning as per
15. $Upload\ all\ training\ data\ as\ Test\ Data$ // Uploading the test data
16. $Classify\ using\ Trained_{Architecture}$// Classify the test data as per stored trained value

Propagation Iterations	100-500
Hidden Neuron Count	20-100
Hidden Layer Count	2
Back Propagation Architecture	Levenberg
Satisfaction Criteria	Gradient
Back Propagation Parameter	Mean Squared Error

Table 1: Ordinal Measures of Neural Network.

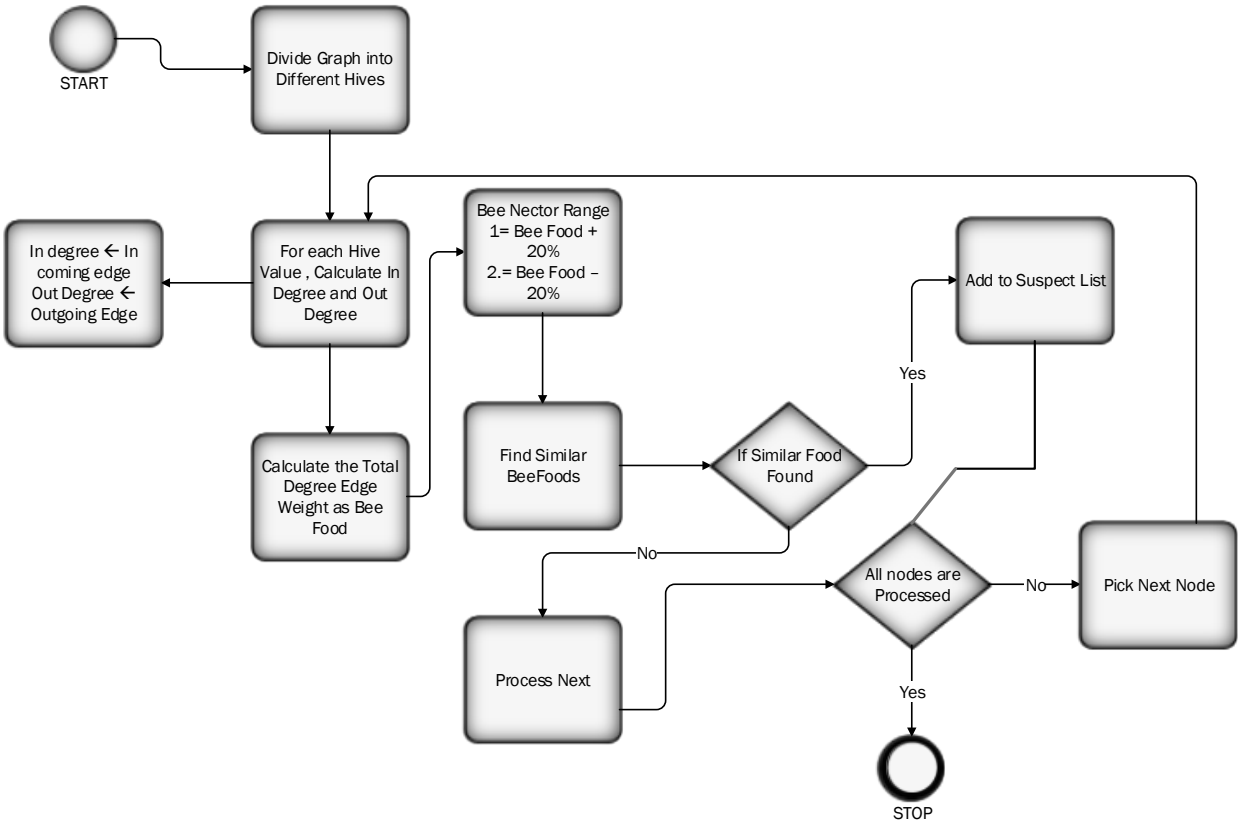

Figure 4: The working architecture of Proposed ABC.

17. *If Classified Label is not similar to the training label* // If the classified value is not similar to trained label
18. $Matched_{Pattern}++$ // The architecture is similar to other architecture
19. *End Function*

The learning and classification mechanism is represented in Figure 5. The flow chart is labeled from 1-10 as per its process occurrence.

Based on the proposed algorithm, the evaluated results are discussed in Section 5.

3 Results

The performance of the proposed subgraph mining approach is evaluated in terms of time performance, memory overhead and number of subgraphs pruned with variation in the supported threshold frequency. The supported threshold is varied to investigate its effect in returning a non-empty set of patterns or subgraphs.

Time performance of the proposed work is evaluated against the four existing studies namely, Ingalalli et al., 2018, Qiao et al., 2018, Abdelhamid et al., 2017, Elseidy et al.,2014 and Le et al., 2020. The considered studies have proposed subgraph pruning strategies inspired by Elseidy et al., 2014 work, who had proposed GraMi for subgraph from larger complex graphs based on the supported threshold frequency. Ingalalli et al., 2018 had proposed MuGraM as an algorithm to identify frequent subgraph patterns from multigraph structure. Qiao et al., 2018 has proposed SSiGraM as a parallel subgraph mining algorithm that was based on Apache Spark framework. Abdelhamid et al., 2017 proposed IncGM+ as a fast incremental system for frequent subgraph mining to resolve the challenges of evolving graphs. Le et al., 2020 developed a Weighted Graph Mining algorithm for subgraph pruning that was named as WeGraMi. In this approach, the weighted graph mining was followed by search space analytics for subgraph pruning.

The experiments are conducted for 10 frequency thresholds that are plotted on X-axis against the running time on Y-axis to evaluate the effectiveness of the proposed work as shown in Figure 6. It is observed that original GraMi required highest running time; however, MuGraM, SSiGraM, IncGM+, WeGraMi including proposed work involved lower running time over different supported thresholds. Further it is also established that the proposed work exhibited the lowest time for subgraph mining on the threshold values under study. This establishes the fact that the proposed work not only outperformed the GraMi but also proved to be better than most of the existing works that were inspired by GraMi.

In addition to running time, memory consumption is another important parameter that decides the feasibility of the proposed technique. Figure 7 compares the memory overhead of the proposed work with IncGM+ and WeGraMi over the supported threshold frequency. It is observed that with decrease in the threshold, the memory consumption rises for all the works. However, this trend is very gradual in case of proposed work. Overall, minimum memory usage is found for the proposed work in comparison to IncGM+ and WeGrami.

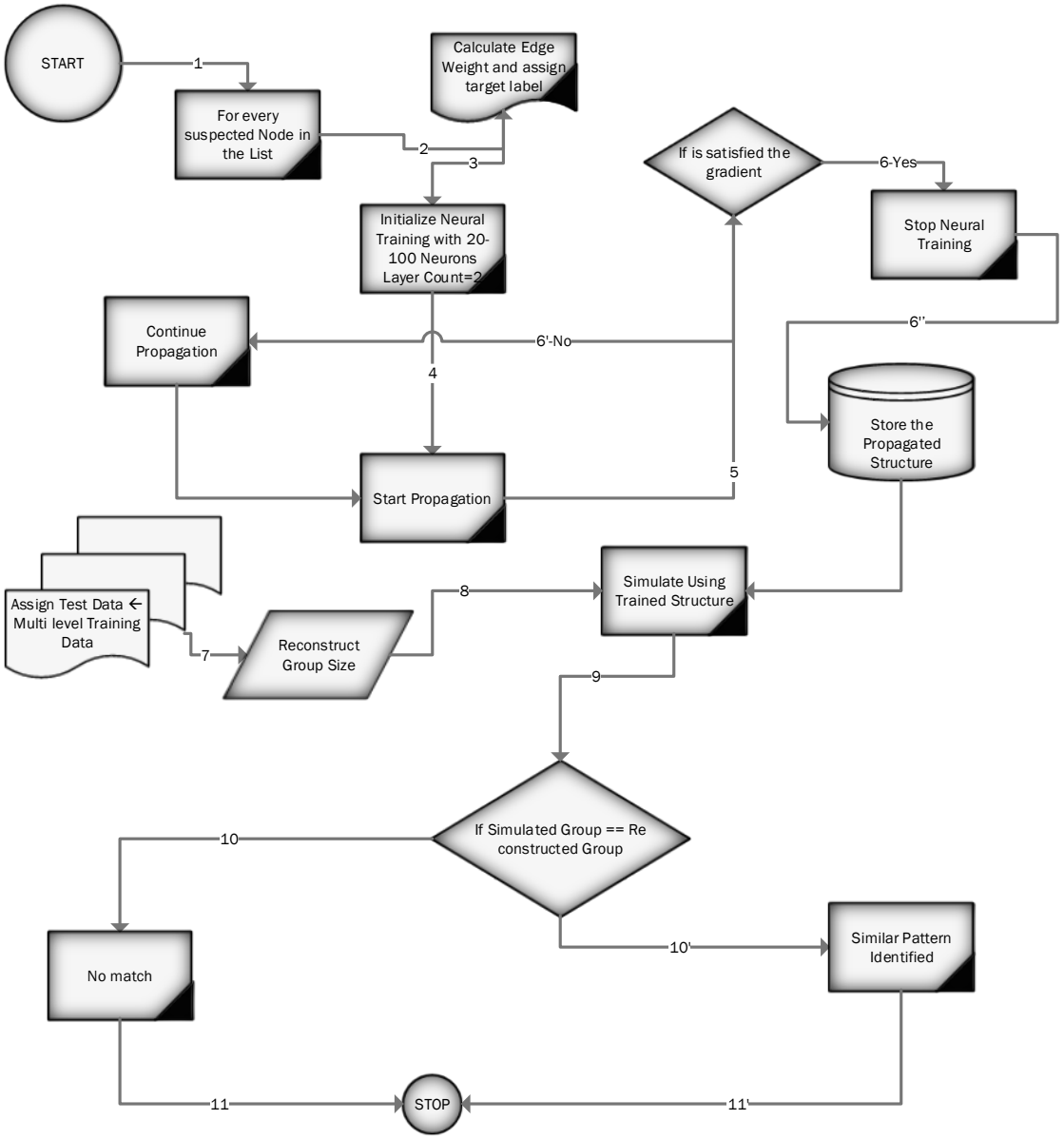

Figure 5: Process Diagram of Neural Networks.

Figure 8 compares the number of subgraphs pruned using various approaches. In addition to above evaluations, the performance of the proposed work is also estimated using quality parameters in terms of True Detection Rate (TDR), False Detection Rate (FDR) and F-Measure in comparison to GraMi. The parametric values are calculated using as follows:

$$TDR = \frac{Total_{true_{detections}}}{Total\ Number\ of\ Detections} \quad (1)$$

Where, TDR is the ratio of the total number of true matchings to the total number of detections.

$$FDR = \frac{Total_{False_{Detections}}}{Total\ Number\ of\ Detections} \quad (2)$$

Where, FDR is the total number of false detections observed to the total number of detections.

$$F - Measure = \frac{2*TDR*FDR}{TDR+FDR} \quad (3)$$

Where, $F - Measure$ is twice the product of TDR and FDR to the summed-up value of TDR and FDR.

Table 2 and Table 3 summarises the analysis of the data for precision, TDR, FDR, and f-measure. Precision and f-measure values observed for both GraMi and proposed work are listed in Table 2. The range of nodes for evaluation lies from 100 to 5000.The parametric values observed for precision calculation are plotted in Figure 9. The parametric values of precision are plotted against a number of nodes from 100 to 5000. It is observed that GraMi achieved an average precision of 66.76%, whereas the average precision of the proposed work is 75.11%. Overall, the proposed work achieved an enhanced precision of 8.35%. F-measure values are compared in Figure 10. It is observed that f-measure for GraMi lies in the range of 0.633 to 0.645, and for proposed work, it lies in the range of 0.721 to 0.751. An average f-measure observed for GraMi and proposed work is 0.65 and 0.74 respectively.

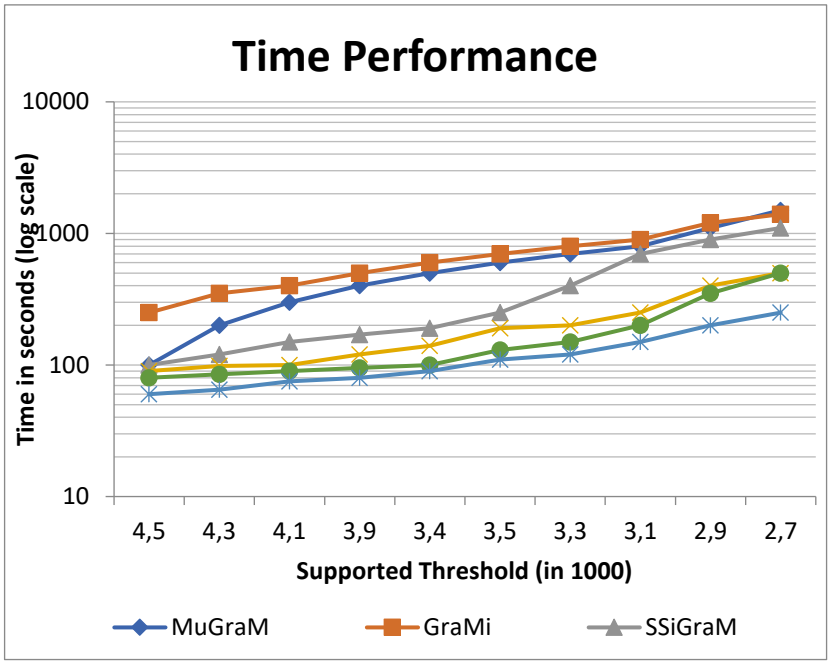

Figure 6: Comparison of Time Performance.

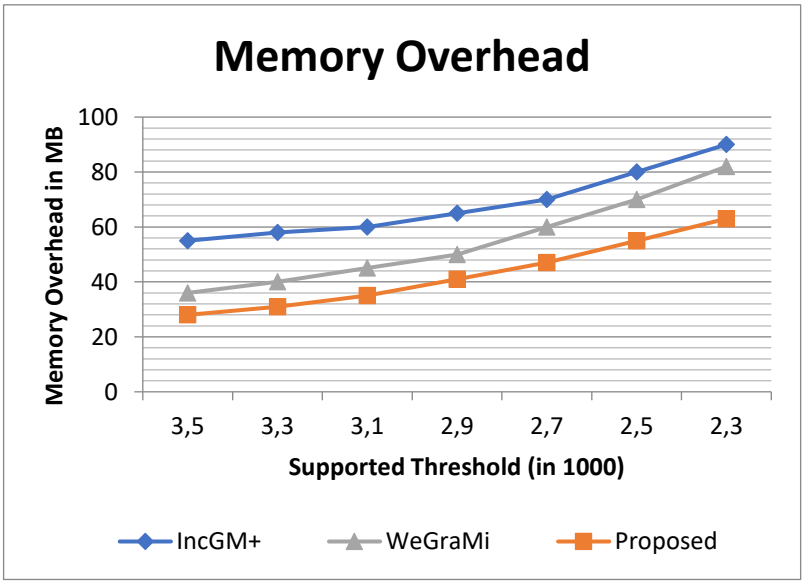

Figure 7: Comparison of Memory Overhead.

F-measure can be understood in terms of the harmonic mean of precision and TDR. It is observed that both values are higher for the proposed work as compared to the GraMi. Therefore f-measure is also higher for the proposed work. On average, there are 0.0915 differences in the f-measure values between the two works. True Detection Rates and False Detection Rates for GraMi and Proposed work are summarized in Table 3. TDR values for GraMi and proposed work are listed in columns 2 and 3 while FDR values of GraMi and proposed work are listed in column 4 and column 5. The numbers of nodes are in the range from 100 to 5000.

TDR of the GraMi and proposed work are compared in Figure 11. The parametric values of TDR are plotted on Y-axis against the number of nodes plotted on the X-axis. GraMi achieved an average TDR of 0.624 as compared to an average TDR of 0.722 for the proposed work. On average, it is concluded that the proposed work had 9% better TDR as compared to the GraMi.

FDR observed for GraMi and proposed work are comparatively plotted in Figure 12. The graph shows that the proposed work demonstrates comparatively low FDR as compared to GraMi. On average, FDR of 0.3324 and 0.2488 is observed for GraMi and proposed work respectively. In other words, the proposed work achieved an average lower FDR of 8.35%.

4 Conclusion

The paper has addressed the challenges faced by subgraph pattern mining of larger network graphs. The authors had designed and evaluated the performance of the proposed

Number of Nodes	Precision		F-measure	
	GraMi	Proposed	GraMi	Proposed
100	0.6571	0.7315	0.633	0.721
500	0.6598	0.7414	0.635	0.726
1000	0.6642	0.7465	0.642	0.73
2000	0.6689	0.7546	0.65	0.738
3000	0.6711	0.7587	0.653	0.743
4000	0.6734	0.7599	0.657	0.747
5000	0.6787	0.7654	0.645	0.751

Table 2: Precision and f-measure for both the datasets.

Number of Nodes	TDR		FDR	
	GraMi	Proposed	GraMi	Proposed
100	0.6101	0.7089	0.3429	0.2685
500	0.6111	0.7102	0.3402	0.2586
1000	0.6211	0.7141	0.3358	0.2535
2000	0.6314	0.7214	0.3311	0.2454
3000	0.6354	0.7276	0.3289	0.2413
4000	0.6412	0.7329	0.3266	0.2401
5000	0.6144	0.7356	0.3213	0.2346

Table 3: TDR and FDR for both the datasets.

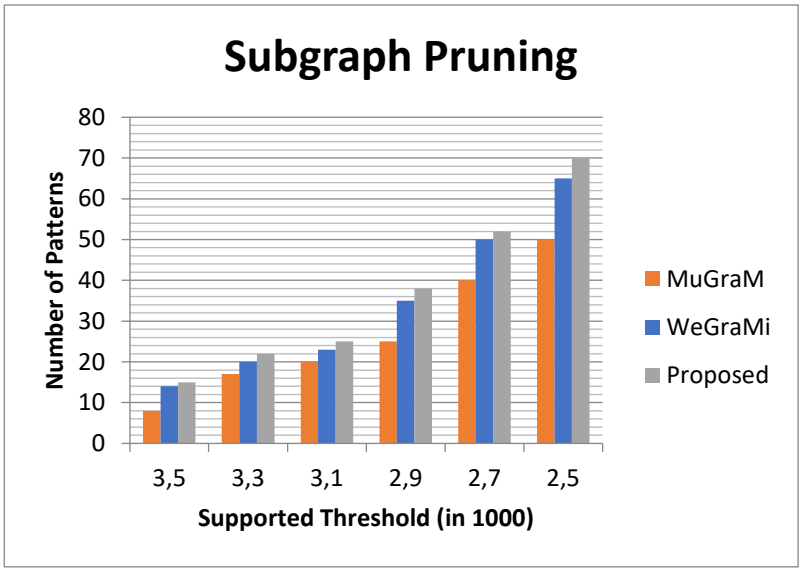

Figure 8: Comparison of subgraph pruning.

structure, which is a combination of Swarm Intelligence and Machine Learning for pattern mining. A new fitness function and a inter and intra hive behavior are introduced for Artificial bee Colony and are cross-validated by Machine learning based Feed Forward Back Propagation Neural Network. The performance of the proposed work is evaluated in terms of TDR, FDR, precision, and f-measure. A range from 100 to 5000 nodes are being analyzed for both proposed and GraMi. It is observed that both proposed work and GraMi achieve an average precision of 75.114% and 66.76%, TDR of 0.7215 and 0.6225, FDR of 0.2488 and 0.3324, and f-measure of 0.736 and 0.645. It is observed that an improved average precision, TDR, FDR, and f-measure of 8.35%, 9.8%, 8.35%, and 9.15% have been demonstrated by the proposed work in comparison to the GraMi. Hence, it is concluded that the proposed work outperformed the existing work.

References

[1] Chen, Y., Zhao, X., Lin, X., Wang, Y. and Guo, D., 2018. Efficient Mining of Frequent Patterns on Uncertain Graphs. IEEE Transactions on Knowledge and Data Engineering, 31(2), pp.287-300. https://doi.org/10.1109/tkde.2018.2830336

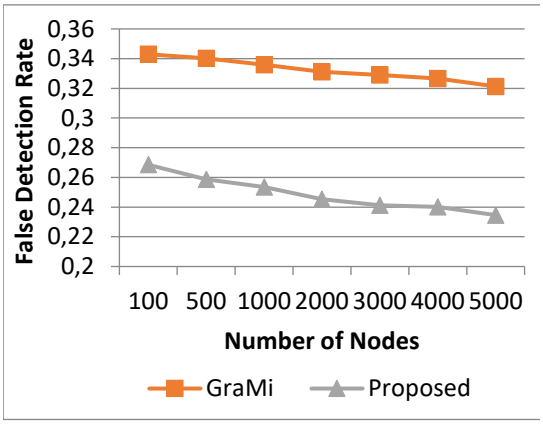

Figure 10: False Detection Rate.

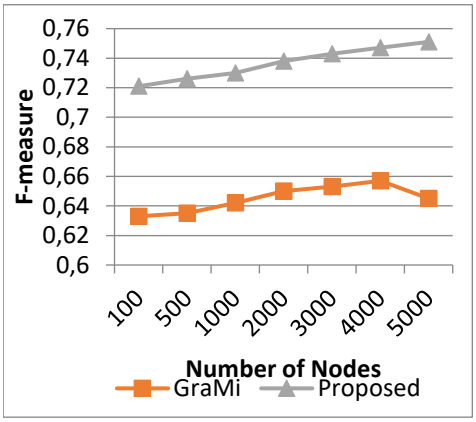

Figure 9: F-measure.

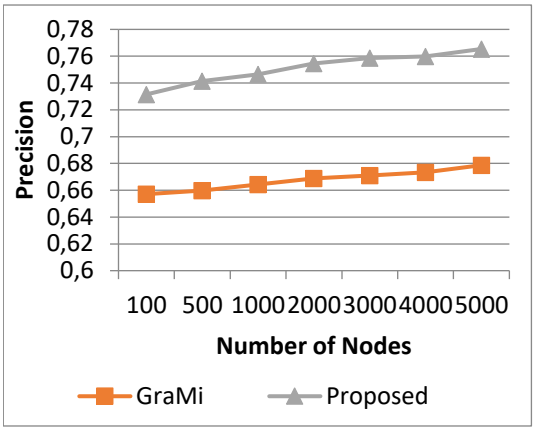

Figure 12: Precision.

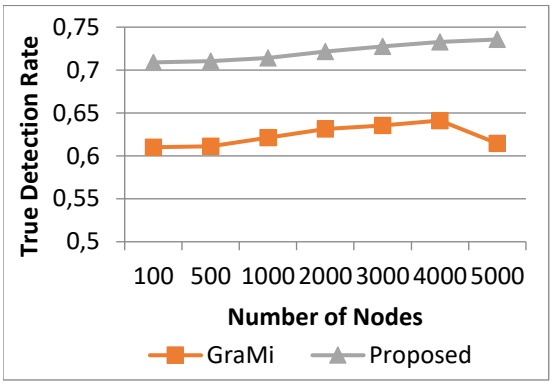

Figure 11: True Detection Rate

[2] Elseidy, M., Abdelhamid, E., Skiadopoulos, S., & Kalnis, P. (2014). Grami: Frequent subgraph and pattern mining in a single large graph. Proceedings of the VLDB Endowment, 7(7), 517-528. https://doi.org/10.14778/2732286.2732289

[3] Ingalalli, V., Ienco, D. and Poncelet, P., 2018. Mining frequent subgraphs in multigraphs. Information Sciences, 451, pp.50-66. https://doi.org/10.1016/j.ins.2018.04.001

[4] Cheng, H., Yan, X., & Han, J. (2014). Mining graph patterns. In Frequent pattern mining (pp. 307-338). Springer, Cham. https://doi.org/10.1007/978-3-319-07821-2_13

[5] Gu, Y., Gao, C., Wang, L., & Yu, G. (2016). Subgraph similarity maximal all-matching over a large uncertain graph. World Wide Web, 19(5), 755-782. https://doi.org/10.1007/s11280-015-0358-9

[6] Yuan, Y., Wang, G., Chen, L., & Wang, H. (2012). Efficient subgraph similarity search on large probabilistic graph databases. Proceedings of the VLDB Endowment, 5(9), 800-811. https://doi.org/10.14778/2311906.2311908

[7] Li, J., Zou, Z., &Gao, H. (2012). Mining frequent subgraphs over uncertain graph databases under probabilistic semantics. The VLDB Journal, 21(6), 753-777. https://doi.org/10.1007/s00778-012-0268-8

[8] Rehman, S.U., Asghar, S. and Fong, S.J., 2018. Optimized and Frequent Subgraphs: How Are They Related? IEEE Access, 6, pp.37237-37249. https://doi.org/10.1109/access.2018.2846604

[9] Abdelhamid, E., Canim, M., Sadoghi, M., Bhattacharjee, B., Chang, Y. C., &Kalnis, P. (2017). Incremental frequent subgraph mining on large evolving graphs. IEEE Transactions on Knowledge and Data Engineering, 29(12), 2710-2723. https://doi.org/10.1109/tkde.2017.2743075

[10] Bhuiyan, M. and Hasan, M.A. (2015) An iterative mapreduce based frequent subgraph mining algorithm. IEEE Trans. Knowl. Data Eng., 27, 608–620. https://doi.org/10.1109/tkde.2014.2345408

[11] Zhao, X., Chen, Y., Xiao, C., Ishikawa, Y. and Tang, J., 2016. Frequent subgraph mining based on Pregel. The Computer Journal, 59(8), pp.1113-1128. https://doi.org/10.1093/comjnl/bxv118

[12] Talukder, N. and Zaki, M.J., 2016. A distributed approach for graph mining in massive networks. Data Mining and Knowledge Discovery, 30(5), pp.1024-1052. https://doi.org/10.1007/s10618-016-0466-x

[13] Aridhi, S., & Nguifo, E. M. (2016). Big graph mining: Frameworks and techniques. Big Data Research, 6, 1-10. https://doi.org/10.1016/j.bdr.2016.07.002

[14] Moussaoui, M., Zaghdoud, M. and Akaichi, J., 2018. A New Framework of Frequent Uncertain Subgraph

Mining. Procedia Computer Science, 126, pp.413-422.
https://doi.org/10.1016/j.procs.2018.07.275

[15] Jena, B., Khan, C., & Sunderraman, R. (2018, November). SparkFSM: A Highly Scalable Frequent Subgraph Mining Approach using Apache Spark. In 2018 IEEE International Conference on Data Mining Workshops (ICDMW) (pp. 990-997). IEEE.
https://doi.org/10.1109/icdmw.2018.00143

[16] Islam, M. A., Ahmed, C. F., Leung, C. K., & Hoi, C. S. (2018, June). WFSM-MaxPWS: an efficient approach for mining weighted frequent subgraphs from edge-weighted graph databases. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 664-676). Springer, Cham.
https://doi.org/10.1007/978-3-319-93040-4_52

[17] Iyer, A. P., Liu, Z., Jin, X., Venkataraman, S., Braverman, V., & Stoica, I. (2018). {ASAP}: Fast, Approximate Graph Pattern Mining at Scale. In 13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18) (pp. 745-761).

[18] Choi, H., Kim, J., Yoon, Y. and Moon, B.R., 2019. Investigation of incremental hybrid genetic algorithm with subgraph isomorphism problem. Swarm and Evolutionary Computation, 49, pp.75-86.
https://doi.org/10.1016/j.swevo.2019.05.004

[19] Preti, G., Lissandrini, M., Mottin, D., & Velegrakis, Y. (2019). Mining patterns in graphs with multiple weights. Distributed and Parallel Databases, 1-39.
https://doi.org/10.1007/s10619-019-07259-w

[20] Rao, B., & Mishra, S. (2019). An Approach to Detect Patterns (Subgraphs) with Edge Weight in Graph Using Graph Mining Techniques. In Computational Intelligence in Data Mining (pp. 807-817). Springer, Singapore.
https://doi.org/10.1007/978-981-10-8055-5_71

[21] Li, L., Zhang, F., & Liu, G. (2019). Multi-fuzzy-objective graph pattern matching with big graph data. Journal of Database Management (JDM), 30(4), 24-40.
https://doi.org/10.4018/jdm.2019100102

[22] Ray, A., Holder, L. B., & Bifet, A. (2019). Efficient frequent subgraph mining on large streaming graphs. Intelligent Data Analysis, 23(1), 103-132.
https://doi.org/10.3233/ida-173705

[23] Priyadarshini, S., & Rodda, S. (2020). Geometric Multi-Way Frequent Subgraph Mining Approach to a Single Large Database. In Smart Intelligent Computing and Applications (pp. 233-244). Springer, Singapore.
https://doi.org/10.1007/978-981-32-9690-9_23

[24] Le, N. T., Vo, B., Nguyen, L. B., Fujita, H., & Le, B. (2020). Mining weighted subgraphs in a single large graph. Information Sciences, 514, 149-165.
https://doi.org/10.1016/j.ins.2019.12.010