IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

USING DUPLICATE REDUCTION TECHNIQUE FOR ANALYSIS, EVALUATION AND IMPROVEMENT OF GRAPH MINING ALGORITHMS.

[1]Miss. Shreya Appa Langote [2]Prof. V.V. Pottigar
[1]M.E. (Computer Science & Engineering) student, [2]Assistant Professor
Department of Computer Science & Engineering
N B Navale Sinhgad College of Engineering, Solapur
Solapur University, Solapur, India

Abstract: At the center of graph mining lies free extension of bases where a base or subgraph autonomously develops into various bigger bases in every cycle. Such a free development, perpetually, prompts the age of copies. Within the sight of chart segments, copies are created both inside and across allotments. Wiping out these copies brings about age and capacity cost as well as extra calculation for its end. Essential point is to plan methods to decrease creating copy foundations as we show that they can't be dispensed with. Proposed philosophy presents three requirement based enhancement strategies, each essentially improving the general mining cost by diminishing the quantity of copies produced. These options give adaptability to pick the correct procedure dependent on chart properties. Proposed philosophy tests show noteworthy advantages of these imperatives regarding capacity, calculation, and correspondence cost (explicit to divided methodologies) across graphs with differed qualities.

Index Terms - **Graph mining, substructure discovery, constraint-based heuristics, duplicates reduction**

I. INTRODUCTION

Substructure discovery is that the process of discovering substructure(s) (a connected subgraph) during a graph that best characterizes an idea embedded therein graph supported some criteria (frequency, compressibility etc.) Many approaches for substructure discovery are proposed within the literature. Main memory based, disk-based [8], [11] and database-oriented approaches [13] address substructure discovery on one machine. With graphs that overwhelm main memory, partition based approaches to substructure discovery have also been proposed.

All of the above-mentioned approaches use an iterative algorithm that: generates all substructures of accelerating sizes (starting from substructure of size one that has one edge), counts the amount of distinct identical (or similar) substructures and applies a metric (e.g., frequency, Minimum Description Length, minimum support etc.) to rank them. In each iteration, either all expanded substructures or a subset (using the rank) are carried forward to limit the search space. This process is repeated until a given substructure size is reached or there are not any more substructures to get. This manner of expansion grows each node during a substructure altogether possible ways as separate substructures in each iteration by adding a foothold thereby generating many substructures of subsequent size. This system is mentioned as independent expansion where a substructure is expanded unaware of other expansions within the same iteration. Non independent expansion makes the method sequential, affecting performance and making it not suitable for giant graphs and partition-based approaches.

The unconstrained independent expansion is complete as it guarantees generation of all possible substructures in every iteration. However, as a by-product of independent unconstrained expansion, duplicates are generated when different substructures, in the same iteration, expand into multiple copies of the same (exact) larger substructure. As an instance expands only from connected edges, a line graph can be generated in a minimum of two ways (missing an edge at either end) while a completely connected graph of k-edges can be generated in k ways (from k (k-1)-edge substructures each missing an edge out of k edges). Only one copy of each substructure needs to be preserved. For each substructure of size k being expanded, the number of duplicates varies from 1 to k - 1. Figure 1 shows an example of duplicates on a line graph and a connected graph (with k = 3.) The lower limit of the duplicates generated is equal to double the

number of substructures. The upper limit depends on the node degree or connectivity. Even the lower limit is detrimental as at least double the work needs to be done to generate all duplicates (and most likely more.) Duplicate identification requires either sorting or pair wise comparisons – both of which are expensive for large numbers.

The duplicates once identified need to be removed to ensure correctness, incurring additional computation cost. Preventive techniques to reduce duplicate generation will not only save duplicate generation cost but pruning cost as well. The challenge, therefore, is to augment the unconstrained independent expansion strategy using heuristics which limit the generation of duplicates. Needless to say, these heuristics should be correct (sound and complete) to generate the same results as an unconstrained expansion. Moreover, heuristics cannot use the knowledge of any other substructure and their expansion details and should retain their independent nature. Hence, reduction of duplicates with low overhead seems more pragmatic than their elimination. If we are able to find out information that characterizes a substructure locally, use that to define a constraint that is sound and complete, and is computationally inexpensive to apply, that would satisfy requirements.

Correct expansion generates an enormous set of intermediate substructures. The amount of intermediate results grows exponentially to a particular substructure size before beginning to decrease. Users, on the opposite hand, have an interest in mining patterns following some user-defined parameters. Hence some pruning properties (based on user-defined parameters) are typically applied that limit the search space and use the simplest substructures for further expansion. Therefore, completeness must be guaranteed even in presence of those pruning properties. Proposed technique augment to independent expansion strategy by introducing three heuristics, each reducing the amount of duplicates generated during graph expansion. These heuristics are often seamlessly integrated into most of the graph representations used for the iterative algorithm. The goal isn't only to spot and optimization for significantly improving the value of graph mining, but also confirm it satisfies scalability and speedup requirement (i.e. work on partitioned graphs.)

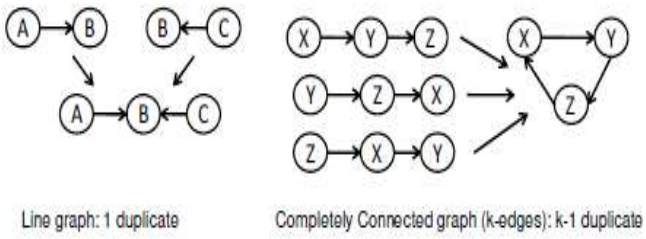

Figure 1. Number of duplicates in graph mining

II. Related Work

In this paper given calculations that utilization a solitary round of guide diminish and can distinguish all occurrences of a given example chart. These calculations are effective both in correspondence cost among mappers and reducers and in the calculation cost at the mappers and reducers. We have not tended to the situation where hubs or potentially edges have names. Neither have we tended to the instance of coordinated charts. A large number of similar methods continue in a clear way. For example, we can in any case express the occurrences of a named, coordinated example chart as an association of CQ's. The auto orphism bunches will in general be littler, so the quantity of CQ's is more prominent, yet similar strategies for assessing CQ's by a multiway join will work. We anticipate that there are provably convertible calculations in all or pretty much every case, except the mapping plans may require some idea. [1]

We built up an equal calculation to produce enormous sans scale systems utilizing the particular connection model. We investigated the reliance idea of the issue in detail that prompted the improvement of a proficient equal calculation for the issue. Different hub dividing plans and their impact on the calculation were talked about too. Our calculation produces systems which carefully observe power-law conveyance. The straight adaptability of our calculation empowers us to create 50 billion edges in only 123 seconds. It will be intriguing to create adaptable equal calculations for other classes of arbitrary systems later on. [2]

Mistake open minded graph coordinating might be an influential idea that has different applications in design acknowledgment and machine vision. Inside the current paper, a substitution separation measure on graphs is proposed. It's bolstered the maximal basic subgraph of two graphs. The new measure is better than alter separation based estimates in that no specific alter tasks close by their expenses got the chance to be characterized. It's officially indicated that the new separation measure might be a measurement. Possible calculations for the effective calculation of the new measure are talked about. [3]

The objective of this paper was to make a basic, parsimonious graph model to portray genuine charts. Our R-MAT model is actually a stage toward this path: we show tentatively that few, assorted genuine charts can be very much approximated by a R-MAT model with the appropriate selection of boundaries. Also, we propose a rundown of regular tests which hold for an assortment of genuine charts: coordinating the force law DGX dispersion for the in-what's more, out-degree; the bounce plot and the distance across of the graph; the solitary worth circulation; the estimations of the rest particular vector (Google-score"); and the stress" dispersion over the edges of the chart. [4]

This paper presents two calculations utilizing the Map/Reduce worldview for mining fascinating and dull examples from an apportioned info graph. A general type of graphs, including coordinated edges and cycles are dealt with by our methodology. Our essential objective is to address versatility, unravel troublesome and computationally costly issues like copy end, authoritative marking and isomorphism location in the Map/Reduce system, without loss of data. Our investigation and examinations show that graphs with a huge number of edges can be dealt with satisfactory speedup by the calculation. [5]

A methodology for powerful assessment of questions indicated over graph databases. The proposed enhancer produces question designs methodically and assesses them utilizing proper cost measurements gathered from the graph database. For now, a graph mining calculation has been adjusted for assessing a given inquiry plan utilizing obliged development. Important metadata relating to the graph database is gathered and utilized for assessing a question plan utilizing a branch and bound calculation. Analyses on various kinds of inquiries more than two chart databases (Internet Movie Database or IMDB and DBLP) are performed to approve our methodology. [6]

To apply social database strategies to bolster visit subgraph mining. A portion of the calculations, for example, copy disposal, standard naming, and isomorphism checking are not direct utilizing SQL. The commitment of this paper is to effectively map complex calculations to social administrators. Not at all like the principle memory partners of FSG, our methodology addresses the most general graph portrayal including different edges between any two vertices, bi-directional edges, and cycles. [7]

A disseminated way to deal with the continuous subgraph mining issue to find fascinating examples with regards to atomic mixes. This issue is portrayed by a profoundly unpredictable hunt tree, whereby no dependable outstanding task at hand forecast is accessible. We depict the three fundamental parts of the proposed circulated calculation, to be specific, a dynamic parcelling of the hunt space, an appropriation procedure dependent on a shared correspondence structure, and a novel receiver initiated load adjusting calculation. The viability of the disseminated strategy has been assessed on the notable National Cancer Foundation's HIV-screening informational collection, where we had the option to demonstrate near direct speedup in a system of workstations. The proposed approach likewise takes into account dynamic asset collection in a non dedicated computational condition. These highlights make it appropriate for huge scope, multi domain, heterogeneous situations, for example, computational networks. [8]

Visit subgraph mining is a functioning examination point in the information mining network. A graph is a general model to speak to information and has been utilized in numerous areas like cheminformatics and bioinformatics. Mining designs from graph databases is trying since chart related activities, for example, subgraph testing; by and large have higher time multifaceted nature than the relating procedure on itemsets, arrangements, and trees, which have been concentrated widely. In this paper, we propose a novel continuous subgraph mining calculation: FFSM, which utilizes a vertical hunt conspire inside a logarithmic graph structure we have created to decrease the quantity of excess up-and-comers proposed. Our exact investigation on manufactured and genuine datasets illustrates that FFSM accomplishes a significant exhibition gain over the current beginning of-the-craftsmanship subgraph mining calculation Span. [9]

Mining progressive subgraphs has pulled it through a various regions, for instance, bioinformatics, web data mining besides, relational associations. There are many promising guideline memory-based techniques available here; anyway they need flexibility as the essential memory is a bottleneck. Taking the colossal data into thought, regular database systems like social databases and thing databases crash and burn pitiably with respect to capability as unending subgraph mining is computationally focused. With the presence of the MapReduce structure by Google, a few experts have applied the MapReduce model on a singular graph for mining visit establishments. In this paper, we propose to make usage of the MapReduce programming model which achieves multifold flexibility on a great deal of named graphs. We attempted our methodology on both veritable and built datasets. This is the essential undertaking to realize trade charts using the MapReduce model. [10]

III. MAPREDUCE

Google's MapReduce is a conveyed model for preparing huge scope information. Clients indicate a guide work and a diminish work. MapReduce takes in a rundown of key-esteem sets, parts them among the conceivable guide errands, and afterward each map work delivers any number of middle of the road key value sets. Sets with comparative keys are assembled at the decrease assignments, and afterward each diminish work performs calculations before yielding qualities, which are either the conclusive outcomes, or perhaps contribution for the following cycle. In a perfect world, MapReduce structures comprise of a few PCs normally alluded to hubs, on the size of tens to thousands. Handling happens on information put away in the file system. Calculation ought to be parallelized over the group, shortcoming lenient, also, planned proficiently. [10]

IV. FREQUENT SUBGRAPH MINING USING MAPREDUCE

We propose utilizing Apache Hadoop1, an open source structure gotten from Google's MapReduce and Google File Framework, to create the continuous subgraphs. Hadoop has become a mainstream approach for circulated and equal registering due its top-level status inside Apache, just as being generally upheld by the network. Calculations through Hadoop are exceptionally adaptable and dependable, making Hadoop an extremely useful asset for preparing huge datasets,
or then again with regards to this paper, huge graph datasets. Utilizing Hadoop iteratively, we can develop all isomorphic subgraphs that surpass a client characterized support. We have two heterogeneous MapReduce employments per emphasis: one for get-together

subgraphs for the development of the people to come of subgraphs, and the other for checking these structures to expel immaterial information. [10]

V. PROPOSED SYSTEM

Proposed approach is different from these existing techniques in terms of the expansion technique. Even in the absence of any prior information on graph characteristics, proposed heuristics significantly reduce duplicates during substructure expansion. Proposed approach offers benefits at several stages of the mining algorithm over existing techniques. First, it reduces the number of duplicates generated at expansion time thereby reducing the amount of intermediate results. Second, it save duplicate removal cost which can be of quadratic complexity in terms of number of similar substructures in the worst case. Third, storage cost of intermediate results is reduced due to non-generation of duplicates. Finally, proposed approach works for both partitioned and non-partitioned mining without any modification. All of these are critical for scaling to arbitrary graph sizes.

In this project we are taking a medical application, in which we are creating graph which contains nodes and edges. Graph nodes represent diseases and its symptoms. Graph edges are in between disease and symptoms.
We are giving input in notepad. In which, Vertex is represented as follows:

Vertex	Vertex_ID	Vertex_Label
V	1	Disease
V	2	BP
V	3	Diabetes
V	4	HighBp

Vertex_ID is always unique for each vertex. And Edge is represented as follows:

Edge	Source_Vertex_ID	Destination_Vertex_ID
D	1	2
D	1	3
D	1	4

In this project we are removing duplicate nodes and edges. Suppose there are two symptom nodes with same name are present for different dieses then we remove the entire node with same name and keep only one node with that symptom name and add the edges to that node. Before running project we represent our graph (diagrammatic representation) of graph using dot language and graphviz software, which shows us diagrammatic representation of graph in terms of nodes and edges. We have developed our project in java, advanced java languages in eclipse IDE. In this we have created website. In this we allowing user to first sign up, Then after creating account user can sing in and upload graph file. This contains vertexes and edges in the format which is mentioned above. Graph must contain duplicate vertexes for ex. Some symptoms are common for two or more diseases so there are two same nodes of symptoms for different diseases. After giving input file, in our project we are extracting file and creating vertexes and edges as given in input file. Before removing duplicates we calculate memory occupied by graph(vertex and edges). After that we are removing duplicates in graph. Then after removing duplicates from graph we again calculate memory occupied by graph. And then we calculate difference between them.

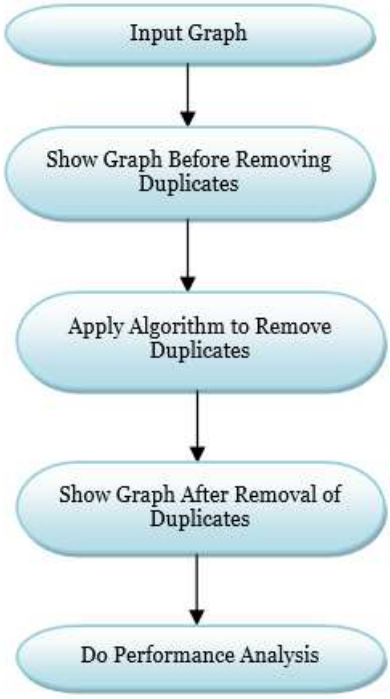

Fig: Proposed Methodology

Algorithm To Remove Duplicates From Graph:

```
1:  begin
2:   for i=0 to vertex_list.size() step 1
3:    begin
4:        for j=i+1 to vertex_list.size step 1
5:            if vertex_label(i) == vertex_label(j) then
6:               begin
7:                 for k=0 to edge_list.size step 1
8:                     Edge e = edge_list.get(k)
9:                     if Source_vertex_id(e) == vertex_id(j) then
10:                        set Source_vertex_id of edge e = vertex_id(i)
11:                    end if
12:                    if Destination_vertex_id(e) == vertex_id(j) then
13:                        set Destination_vertex_id of edge e = vertex_id(i)
14:                    end if
15:               end for
16:               Delete node from vertex_list(j)
17:               Comment: Deleting duplicate node from graph
18:           end if
19:    end for
20:  end for
```

Memory Before Removing Duplicates: 45296632 bytes
Memory After Removing Duplicates: 45520584 bytes
Difference: 223952 bytes

In above memory calculation if we see there is major difference in memory occupied by graph before removing duplicate nodes and memory occupied by graph after removing duplicate nodes. Because of this reduction in memory storage cost is also get reduced. Suppose if we are performing activity on graph like in this medical application by giving symptom we can get diseases occurred by this symptom. So, before removing duplicate nodes we have to search for all disease destination node for that symptom(because nodes are repeated for various symptoms) but, after removing duplicates there is only one node of that symptom so if we give destination nodes i.e. dieses nodes. After removing duplicate nodes computation time is also get reduced.

After removing duplicates from graph we create new graph file whose syntax is same as input file which contains new created graph without duplicates. (User can download this file after giving input on website). Using graphviz software and dot language we again represent graph without duplicates.

VI. Heuristics For Constrained Expansion

Independent substructure expansion in each iteration of graph mining generates duplicates. Independent expansion by definition cannot use any additional information. Hence, all substructures need to be expanded in all possible ways. Further note that no duplicates are generated when a single substructure is expanded. Duplicates are generated by different substructure during their expansion and this is determined by the degree of nodes and the connectivity of the substructure being expanded. Hence, duplicates cannot be avoided by independent expansion. Replacing independent expansion by using different expansion techniques to avoid duplicate generation requires bookkeeping and sequentializes the entire process further making it unsuitable for a distributed paradigm (as you cannot compare with duplicates generated in different partitions.) With proposed approach on adapting graph mining to distributed framework, independent expansion cannot be avoided.

Identification of heuristics which can be applied locally to each instance to reduce duplicates and exhibit completeness globally is needed. For any substructure representation, the local information available are: vertex ids, vertex labels, and edge labels.
As defined earlier, an edge consists of five elements: edge label, connecting vertex labels and their vertex ids. Each vertex has a unique id while labels (both vertex and edge) are not necessarily unique. Note that for the same substructures to occur multiple times in a graph, it is mandatory for edge and vertex labels to repeat.

VII. Conclusion

Scalability has been the main concern in graph mining and has been, typically, addressed and overcome by using different approaches and architectures to an equivalent algorithm. Proposed technique takes such an approach in identifying the effect of duplicates on the performance of graph mining algorithms. Supported that observation, it proposes variety of heuristics to scale back the amount of duplicates generated to significantly improve the performance of those algorithms.

Acknowledgment

I profoundly grateful to Prof. V.V. Pottigar for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. I would like to express my deepest appreciation towards Principal Dr. S.D. Navale, Prof. A.A. Phatak HOD department of computer engineering and PG coordinator Prof. B.R. Solunke. I must express my sincere heartfelt gratitude to all staff members of computer engineering department who helped me directly or indirectly during this course of work. Finally, I would like to thank my family and friends, for their precious support.

References

[1] Foto N. Afrati, Dimitris Fotakis, and Jeffrey D. Ullman. Enumerating subgraph instances using map-reduce. Technicalreport, Stanford University, December 2011.

[2] Md. Maksudul Alam, Maleq Khan, and Madhav V. Marathe. Distributed-memory parallel algorithms for generating massive scale-free networks using preferential attachment model. In SC, page 91, 2013.

[3] Horst Bunke and Kim Shearer. A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters, 19:255–259, 1998

[4] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-MAT: A recursive model for graph mining. In SIAM, Florida, USA, April 22-24, 2004, pages 442–446, 2004.

[5] Sharma Chakravarthy and Subhesh Pradhan. DB-FSG: An SQL-Based Approach for Frequent Subgraph Mining. In DEXA, pages 684–692, 2008.

[6] Soumyava Das and Sharma Chakravarthy. Partition and conquer: Map/reduce way of substructure discovery. In DaWaK 2015, Valencia, Spain, September 1-4, 2015, pages 365–378, 2015.

[7] Soumyava Das, Ankur Goyal, and Sharma Chakravarthy. Plan before you execute: A cost-based query optimizer for attributed graph databases. In DaWaK 2016, Porto, Portugal, September 6-8, 2016, pages 314–328, 2016.

[8] Mukund Deshpande, Michihiro Kuramochi, and George Karypis. Frequent Sub-Structure-Based Approaches for Classifying Chemical Compounds. In IEEE International Conference on Data Mining, pages 35–42, 2003.

[9] Giuseppe Di Fatta and Michael R. Berthold. Dynamic load balancing for the distributed mining of molecular structures. volume 17, pages 773–785, 2006.

[10] Steven Hill, Bismita Srichandan, and Rajshekhar Sunderraman. An iterative mapreduce approach to frequent subgraph mining in biological datasets. In Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine, BCB '12, pages 661–666, 2012.

[11] Lawrence B. Holder, Diane J. Cook, and Surnjani Djoko. Substucture Discovery in the SUBDUE System. In Knowledge Discovery and Data Mining, pages 169–180, 1994

[12] Jun Huan, Wei Wang, and Jan Prins. Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism. ICDM '03, pages 549–552, Washington, DC, USA, 2003.

[13] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In Principles of Data Mining and Knowledge Discovery, pages 13–23, 2000.

[14] George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. In JPDC, volume 48, pages 96–129. Elsevier, 1998.

[15] Arijit Khan, Francesco Bonchi, Aristides Gionis, and Francesco Gullo. Fast reliability search in uncertain graphs. In Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014., pages 535–546, 2014.

[16] Michihiro Kuramochi and George Karypis. Frequent Subgraph Discovery. In IEEE International Conference on Data Mining, pages 313–320, 2001.

[17] W Lin, X Xiao, and G Ghinita. Large-scale frequent subgraph mining in mapreduce. In ICDE, pages 844–855, 2014.

[18] Y Liu, X Jiang, H Chen, J Ma, and X Zhang. MapReduce- Based Pattern Finding Algorithm Applied in Motif Detectionfor Prescription Compatibility Network. In Advanced Parallel Programming Technologies, pages 341–355, 2009.

[19] Wei Lu, Gang Chen, Anthony K. H. Tung, and Feng Zhao. Efficiently extracting frequent subgraphs using mapreduce. In Proceedings of the 2013 IEEE International Conference on Big Data, 6-9 October 2013, Santa Clara, CA, USA, pages 639–647.