Continuous Join between Dynamic Query and Graph Stream

Ms.V. G. Powar[1] Mr.P.C.Bhaskar[2]
[1]Departement of Computer Science and Technology [2]Departement of Electronics and Technology
[1,2]Department of Technology, Shivaji University, Kolhapur

Abstract— Graph is widely used in various real applications such as social network modeling and chemical compound analysis. In many applications, graphs are often evolving along the time in a stream fashion instead of remaining static. These evolving graphs can be modeled as graph stream. The available work is proposed on static subgraph search. We proposed method to continuously join query stream and graph stream. Nodes neighborhood information is used as filtering feature to minimize candidates for join. Nearest Neighbor Trees are generated using neighborhood information. To minimize execution cost nearest neighbor trees are converted to numerical vectors. Proposed algorithms use Dominant vector and skyline join methods for stream join. Here we present comparative analysis on these two methods.

Key words: NNT, NPVs, Dominant Vector Join Algorithm

I. INTRODUCTION

Data sources such as social media streams and network traffic can be modeled as multi-relational graphs. Such graphs are defined by a set of relations between a potentially diverse set of entities. For example, a social media data stream contains a diverse set of entity types such as person, movie and image, and relations such as friendship or liking of a post or image. For cyber-security, a network traffic data source can be represented as a graph with IP address as nodes.

From a database perspective, real-time querying of streaming data is important factor while continuously joining graph. Subgraph search is an effective method to find out substructures present in the graph. Graphs grow over a time in graph stream, so it is important to get real time response while finding match between graph streams.

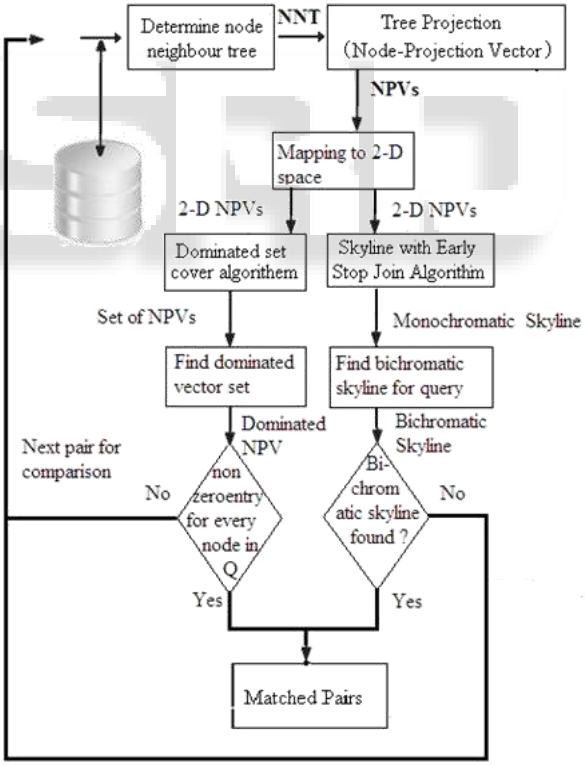
Existing work done for subgraph search include closure tree[1], GraphGrep[2],TALE[3], Gcoding[4] and is meant for static graph data. Closure tree uses maximum match between vertices and needs more iterations. Graphgrep uses maximum path length to filter the graphs as candidate pair, and results in many false positive pairs. TALE uses hashing method to store node neighborhood information. Gcoding assigns a signature to each vertex based on its local structures. A spectral graph code is generated by combining all vertex signatures in a graph. A necessary condition for subgraph isomorphism was derived based on spectral graph codes. In [5], algorithms are proposed to find match between static query pattern and a graph stream, using numerical vectors derived for the graph stream and query pattern. In some applications query pattern may also be dynamic in nature, for example friend list in social networks if assumed as query pattern.

So to deal with dynamic query and graph stream, we proposed dominant vector and skyline join algorithms. It uses node neighbor tree (NNT) [5] for preprocessing of the graph, where NNT represents edge structure for every node in the graph. NNTs are converted to numerical projection vectors (NPV) [5] using dimensions defined for the graph; dimensions show list of edges at each level of the graph. Using numerical vectors, dominant relationship and skyline are find out to join to graph stream.

II. PROPOSED ALGORITHM

System modules implemented to the completion of system are defined below. System is implemented around input domain of graph database. Graph stream, both query and search are generated as actual input to the system. System report pair of matching query stream and graph stream as output. Fig. 1 shows architecture of the system. Graphs are preprocessed to generate NNTs and NPVs for both the graph streams.

NPVs are processed to find dominant vector for query graph against graph stream for every graph change in either of the stream in dominated vector algorithm. In skyline join algorithm monochromatic and bichromatic skyline [6] for the query stream is checked across the graph stream.



A.
Fig. 1: System Architecture

A. Graph Preprocessing

Graphs are preprocessed on the basis of their structural information, to filter out negative pairs while joining the graph streams.

1) Construction of NNT:

While building a graph all simple paths upto the depth of graph are traversed and added to the NNT. NNT is build for every node in the graph.

Pseudo code for generating NNT is shown below.

```
Procedure Buil_NNT
Input: Graph G
Output: Set of NNTs
Begin:
for each node U of G
NNT(U).root=U;
L=U->left;
R=U->right;
NNT(U)->left= Build_NNT(L);
NNT(U)->right= Build_NNT(R);
NNT_set= NNT_set U NNT(U);
Return NNT_set;
End
```

Fig. 2 shows example graph and Fig. 3 shows its NNTs.

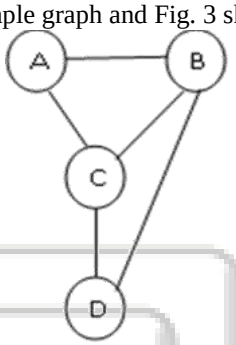


Fig. 2: Example Graph

NNTs drawn for the nodes in Fig. 2 are shown in Fig. 3. Consider node A in graph, it is having edges to nodes C and B at level 1, so these edges are added to NNT(A) in Fig. 3(a). At Level 2 nodes C and B are having edges to B,D and C nodes. Edges CB,CD,BD,BC are added at level 2 as shown in Fig.3(a).

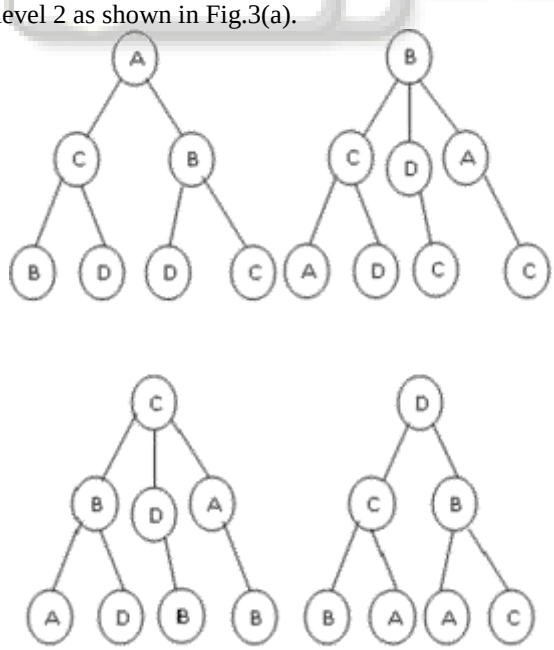


Fig. 3: (a)NNT(A) (b)NNT(B) (c)NNT(C) (d)NNT(B)

2) Projection to NPVs:

To project NNTs to NPVs dimensions of the graph are used. Dimension of graph G is defined as set of triples <l, U, V>. Every triple in dimension represents edge, U to V present at level l in graph G. Fig. 4 shows dimension for graph in Fig. 2. NPV is generated for every NNT by counting no of occurrences of each dimension in the respective NNT. Length of NPV is equal to number of triples in dimension of G.

Fig. 5 shows NPVs for NNTs in Fig. 3

Fig. 4: Dimension of graph G in Fig. 2

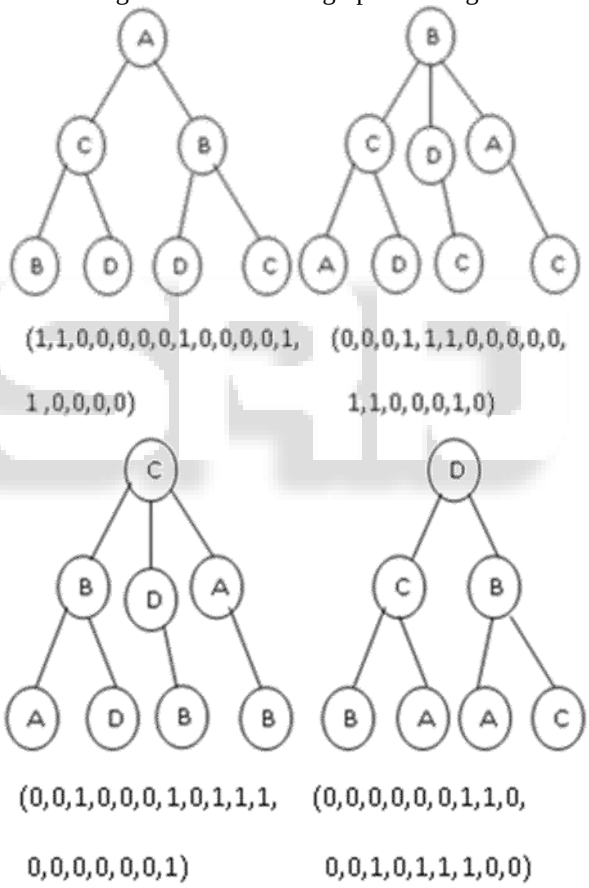


Fig. 5: NPVs for NNTs in Fig.3

3) Mapping of NPVs to 2-D space:

To minimize search space n-length NPVs are converted to 2 dimension vectors i.e. zero dimension and non-zero dimension. These 2-D NPVs are mapped to 2-D space by sorting them on zero and non-zero dimensions. Position counter is calculated for every node of both the streams. Dominant counter is calculated for every node in graph stream.

B. Continuous Join of Query and Graph Stream

Graphs are stored in the form of position counters and dominant counter for further processing. These counters are updated for every graph change operation. Dominant

counter is used to find out dominant vector for query graph with respect to graph in dominant vector join algorithm. Position vectors are used to find skylines of query in skyline join algorithm.

1) Dominant Vector Join Algorithm

Algorithm finds dominant vector for query graph using dominant counter. Dominant counter is m-length vector, where m is number of nodes in query graph. Every element in dominant counter represents number of dimensions of query graph nodes dominated by graph node. Point Y dominates point X if its coordinates are greater than or equal to coordinates of point X. Dominant vector is (n x m) array where, n is number of nodes in graph and m is number of nodes in query. Every i^{th} row of dominant vector is dominant counter for i^{th} node in graph. If all query vector are dominated by the graph nodes, pair of query and graph represents positive result for join.

Pseudo code for Dominant vector algorithm is shown below.

1) Procedure Dominant _Vector_Join
Input: query stream {Q1,Q2,……Qk1}, graph stream {G1,G2,……Gk2}
Output: Positive candidate pairs
Begin:
for i ← 1 to k1 and j ← 1 to k2 do in parallel
for u∈ Qi and v∈ Gj do in parallel
update position counter for u and v
update dominant counter for v
mark query vectors dominated by graph vectors according to dominant counter value
for i ← 1 to k1
if Gj dominates all vectors of Qi
result= result U (i ,j)
return result
End

2) Skyline Join Algorithm

Monochromatic skyline for set of points in space is point with highest position in the space. By using position vector for query nodes monochromatic skyline is defined for the query graph. The skylines nodes of query graph are checked across graph nodes to find bichromatic skyline of query with respect to graph. bichromatic skyline represents points in graph which dominates skyline points of the query. If there exist a bichromatic node, the respective pair of query and graph is excluded from join.

Pseudo code for Skyline Join algorithm is shown below.

1) Procedure Skyline_Join
Input: query stream {Q1,Q2,……Qk1}, graph stream {G1,G2,……Gk2}
Output: Positive candidate pairs
Begin:
for i ← 1 to k1 and j ← 1 to k2 do in parallel
for u∈ Qi and v∈ Gj do in parallel
update position counter for u and v
find monotonic skyline S for Qi.
find Max node D1 and Max node D2 for Gj
if S is bichromatic skyline across D1 and D2
break
else
result= result U (i ,j)
return result
End

III. EXPERIMENTAL RESULTS

Dataset is manually created for creating graphs of phone transaction between individuals. For creating streams timer is used which periodically inserts or deletes node to or from the graphs. Visual studio 2010 software platform is used to perform the experiment. SQL Server 2008 is used as database server.

Results are shown in Fig. 6 for graph of 80 nodesjoined with query of 11 nodes.

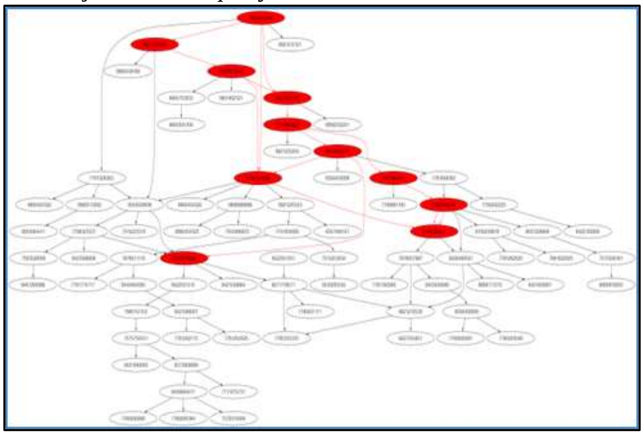


Fig. 6: Positive Result for Dominated Vector Algorithm

Fig. 7 shows dominant vector for the given pair of query and graph in Fig. 6.

2 2 2 2 2 2 2 2 2 2 2	1 2 1 1 1 2 2 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 1 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
1 2 1 1 2 2 2 2 2 2 1	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 1 1 1 1	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 1 1 2 2 2 2 2 2 1	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 1 1 1 1 1 1 1	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 1 1 1 1 1 1	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 1
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 1 1 1 1 1 1 1 1	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2
2 2 2 1 2 2 2 2 2 2 1	2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2	1 2 1 1 1 2 2 2 2 2 1
2 2 2 2 2 2 2 2 2 2 2	1 1 1 1 1 2 2 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2

Fig. 7: Dominant Vector

Fig. 8 shows experimental comparative analysis of the proposed algorithms based on time requirement.

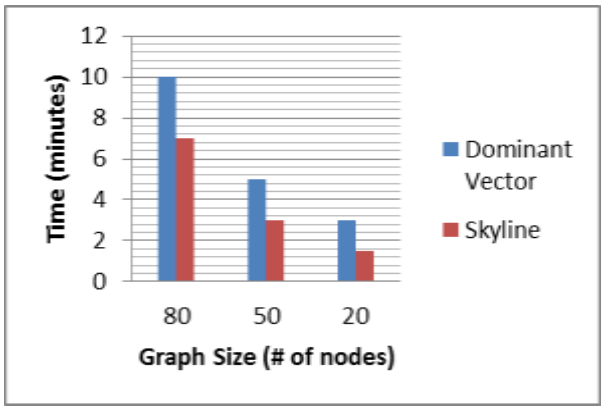


Fig. 8: Comparative Analysis of Algorithms

IV. CONCLUSION

In this system continuous join between query and graph is implemented using node neighbor tree and node projection vector as graph preprocessing technique. It incurs more cost to store and process lengthy NPV for large graphs. So, to minimize cost NPVs are mapped to 2-D space by counting zero and non-zero entries in the original NPVs.

First approach used towards dynamic subgraph search is dominated set cover algorithm. It finds dominated vector set for the dynamic query with respect to graph stream by comparing position of query nodes in the 2-D space. Second approach is based on the skylines. It finds bichromatic skyline of the graph with respect to dynamic query graph.

REFERENCES

[1] H. He and A.K. Singh, "Closure-Tree: An Index Structure for Graph Queries," Proc. 22nd Int'l Conf. Data Eng. (ICDE), 2006
[2] D. Shasha, J.T.L. Wang, and R. Giugno, "Algorithmics and Applications of Tree and Graph Searching," Proc. ACM SIGACT SIGMOD Symp. Principles of Database Systems, 2002.
[3] Y. Tian and J.M. Patel, "Tale: A Tool for Approximate Large Graph Matching," Proc. 24th Int'l Conf. Data Eng. (ICDE), 2008.
[4] L. Zou, L. Chen, J.X. Yu, and Y. Lu, "A Novel Spectral Coding in a Large Graph Database," Proc. Int'l Conf. Extending Database Technology (EDBT), 2008.
[5] Lei Chen, Changliang Wang, "Continuous Subgraph Pattern Search over Certain and Uncertain Graph Streams", IEEE transactions on knowledge and data engineering, vol. 22, no. 8, august 2010
[6] X. Lian and L. Chen, "Monochromatic and Bichromatic Reverse Skyline Search over Uncertain Databases," Proc. ACM SIGMOD,2008.

