



Dynamic Query Search Over Graph Stream

Ms.Vaishali G. Powar¹ and Mr.Pradeep C. Bhaskar²

¹Department of Computer Science and Technology, Shivaji University, Kolhapur, Maharashtra, India

²Department of Electronics and Technology, Shivaji University, Kolhapur, Maharashtra, India

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 find match between query stream and graph stream. Stream comprises sequence of graph at continuous timestamps. Graphs are represented in the form of numerical vectors. Using these numerical vectors, dominance is checked between query stream and graph stream. These methods make use of dominant vector and skyline to reduces number of comparisons made to find match between query stream and graph stream.

Keywords: Query Stream; Graph Stream; Numerical Vector; Dominant Vector; Skyline.

I. Introduction

Many data domains such as chemical data, biological data, social networks and the web are naturally structured as graphs. Graphs may be evolving along the time in a stream fashion instead of remaining static in many applications like social communities, traffic networks. For example, in a traffic network, the links between nodes are changing over the time. These changing graphs can be modeled as graph stream. 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 set and skyline stop join algorithms. It uses node neighbor tree [5] for preprocessing of the graph, where NNT represents edge structure for every node in the graph. NNTs are converted to numerical vectors [5] using dimensions defined for the graph; dimensions show list of edges at each level of the graph. Using numerical vectors dominant relationship is checked to find out the matching candidate pair as a result.

II. Proposed Algorithm

In this section data pre-processing and actual algorithms for dynamic query search over graph stream are described.

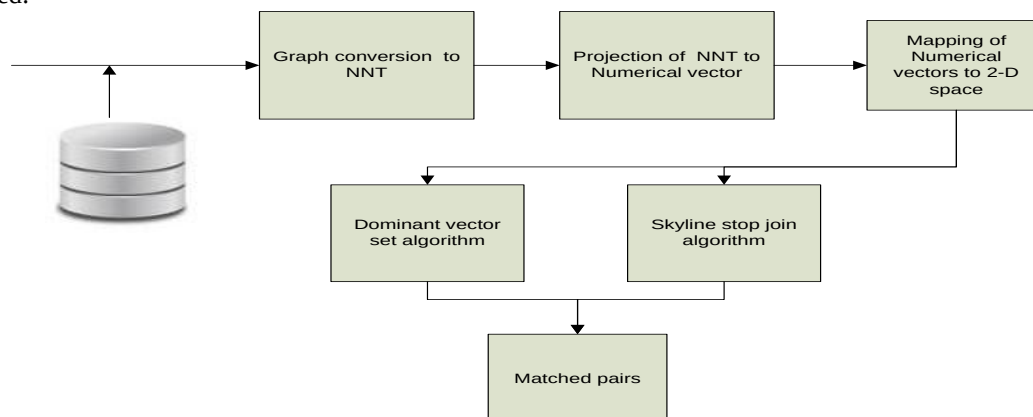


Figure 1: System Architecture

Graph stream and query streams are preprocessed to convert them into nearest neighbor tree (NNT), this NNTs are then converted to the numerical vectors. Finally numerical vectors are used to find out dominant vector and skyline, in dominant vector set algorithm and skyline stop join algorithm.

A Graph preprocessing

To minimize number of computations graphs in stream are continually converted to NNTs. Given a graph G of level l , $NNT(u)$ represents tree rooted at node u with all the simple paths in $NNT(u)$ upto level l .

1. Graph Conversion To NNT

Given a graph $G(V,E)$, $NNT(u)$ shows local structure of vertex u to its direct and indirect neighbors in G up to level l . Fig. 2 shows example graph and its respective NNTs. NNTs are shown in the Fig.3. While building the NNT all simple paths up to depth of the graph from the node are traversed and edges are added to it. For example node A in G has path to A and B at level 1. So these edges are added in NNT T_1 of A as shown in Fig.3a. At level 2 node B is having path to C , so this edge is added to T_1 . NNTs are build for all vertex in the graph.

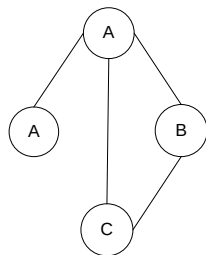


Figure 2: Example Graph

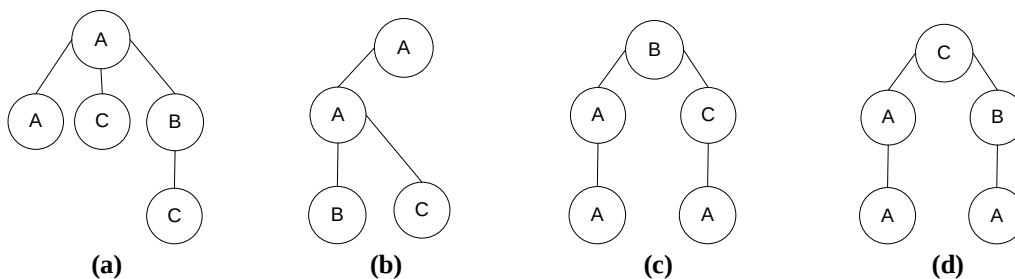


Figure 3: (a) NNT T_1 (A) (b) NNT T_2 (A) (c) NNT T_3 (B) (d) NNT T_4 (C)

2. Projection of NNT to Numerical Vectors

Before projecting NNT to numerical vector dimension for graph G is derived. Dimension is a triple $\langle l, lab1, lab2 \rangle$, where it denotes distinct edge $lab1$ to $lab2$ at level l in G . After finding dimensions for G , NNTs are projected to numerical vectors by counting no of occurrences of each dimension in the respective NNT. For example dimensions for the NNTs in Fig.3 are as shown in Fig. 4a. For every NNT occurrences of the each dimension are counted and count is added to numerical vector. Numerical vectors for NNTs in Fig.3 are shown in Fig.4b.

$$\begin{aligned} &\langle 1, A, A \rangle \langle 1, A, C \rangle \langle 1, A, B \rangle \langle 1, B, A \rangle \langle 1, B, C \rangle \langle 1, C, A \rangle \langle 1, C, B \rangle \\ &\langle 2, B, C \rangle \langle 2, A, B \rangle \langle 2, A, C \rangle \langle 2, A, A \rangle \langle 2, C, A \rangle \langle 2, B, A \rangle \end{aligned}$$

Figure 4 (a): Dimensions of graph in fig.2

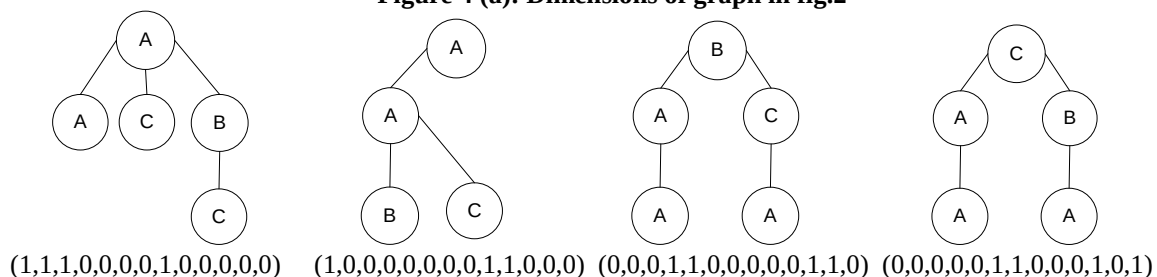


Figure 4 (b): Numerical vectors for NNTs in fig.3.

3. Mapping numerical vectors to 2-dimentional space

To minimize the search space, numerical vectors are mapped to zero and non-zero dimensions. All the nodes are arranged by descending order on their 2 dimensions. After arranging in descending order nodes are mapped to

2-dimensional space. For every node of graph stream 2 counters are observed, a position counter and dominant counter.

B Dynamic Subgraph search over graph stream

Preprocessed data is stored in the form of position counters and dominant counter, and updated continuously for respective updates in the graph stream. Dominant vector cover join algorithms and skyline stop join algorithms are then used to find actual subgraph match.

1. Dominant vector cover join

Dominant vector cover join algorithm find out positions for both query and graph stream from the position counter. Position counter denotes position of the node in 2-D space. It then finds out dominant counter of every node in the graph stream with respect to query nodes. Dominant counter of node u in graph stream is n length vector where, every i^{th} element denotes the number of dimensions dominated by u of node i . Here n is no of nodes in the query graph. If graph dominant vector dominates all query nodes then respective query and graph pair is reported as positive candidate pair. Pseudo code for the algorithm is given below.

Procedure Dominant_vector_cover_Join

Input: query stream $\{Q_1, Q_2, \dots, Q_{k1}\}$, graph stream $\{G_1, G_2, \dots, G_{k2}\}$

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 stop join

Skyline stop join algorithm first finds out monochromatic skyline [6] for query stream, which is node in the query stream in the 2-D space dominating all the query nodes on one of the dimension. Algorithm then find out bichromatic skyline [6] of query stream across the graph stream. Bichromatic skyline is vector set from graph stream which dominates nodes in monochromatic skyline of query stream. Algorithm when find a bichromatic skyline point with respect to graph it reject the respective pair. Otherwise this pair is included in result as a query graph is subgraph of graph stream. Pseudo code for the algorithm is given below.

Procedure skyline stop join

Input: query stream $\{Q_1, Q_2, \dots, Q_{k1}\}$, graph stream $\{G_1, G_2, \dots, G_{k2}\}$

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. Experiments And Results

The test dataset for evaluation of the algorithms is selected from [7]. Dataset contains friend list on social networks. Visual studio 2010 software platform is used to perform the experiment. SQL Server 2008 is used as database server.

The experiments are executed on query and graph with average number of 35 nodes. Stream for both query and graph is formed by inserting timer and implementing autodelete_nodes function in the module. Fig.5a shows dominant counter for the stream graph instant. As it is not having non zero count in entire dominated counter set it results in non matching pair of query and graph. Fig.5b shows bichromatic skyline of query with respect to graph and results in non matching pair.

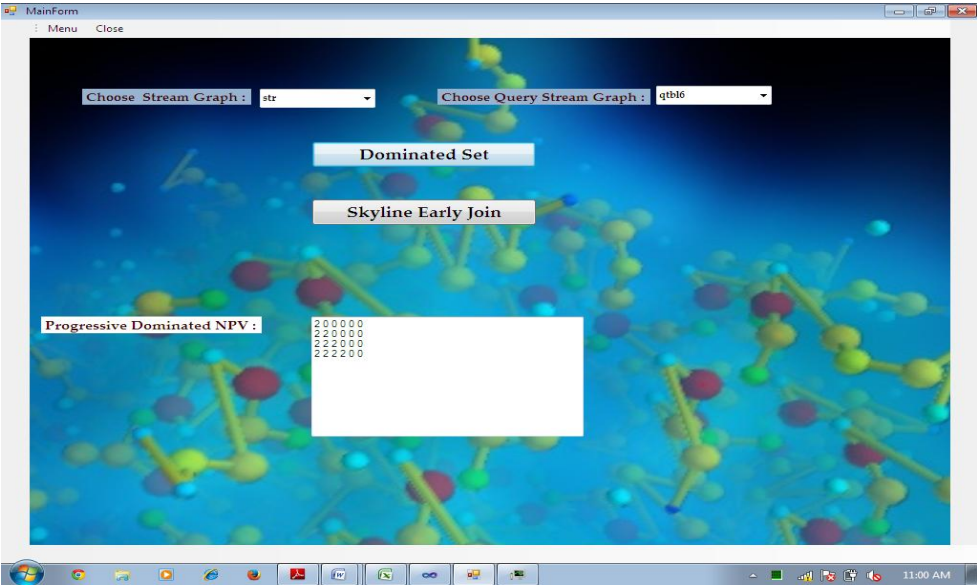


Figure 5 (a): Progressive Dominated counter

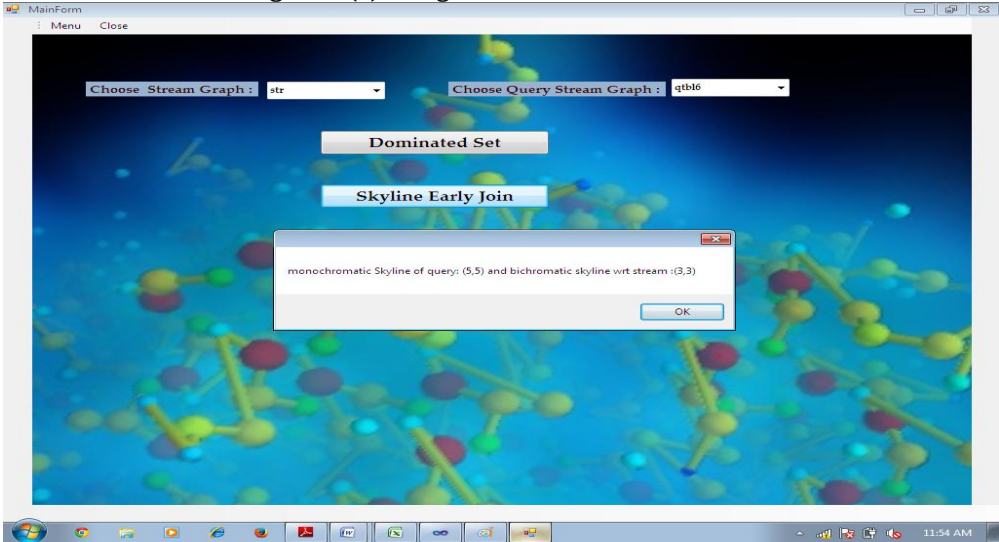


Figure 5 (b): Bichromatic Skyline point

Figure 6 shows experimental analysis of the two algorithms in terms of stream size and time required for subgraph search. Y-axis shows time in minutes and X-axis shows average stream size.

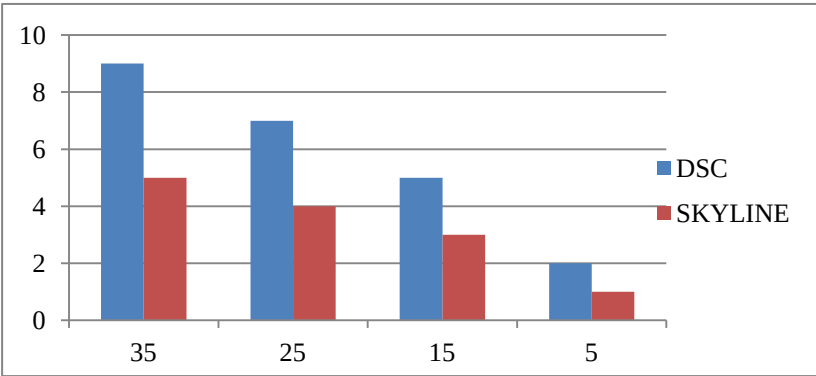


Figure 6: Comparison of time efficiency

IV. Conclusion

Node neighbor tree is used for storing graphs. NNTs are converted to numerical vectors. These numerical vectors are mapped to 2-D space to minimize the search space. Proposed algorithms work with dynamic queries over the graph stream. On graph update, algorithm only updates the position counters reducing pre processing cost. Dominant vector and skylines are used for filtering the subgraph search pairs, which minimizes total number of comparisons. As seen in the time analysis, skyline stop takes time less than dominant vector cover algorithm.

V. 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.
- [7] *Friendster Data Set*, <https://archive.org/details/friendster-dataset-201107>