International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (02), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

Partitioning Directed Acyclic Graphs for path specific analysis

Sushobhit Singh[1], Ajay Kumar Saxena[2]

[1,2]*Department of Electrical Engineering, Dayalbagh Educational Institute, (India)*

ABSTRACT

Graph partitioning is a very well-studied and applied problem in graph theory. Directed Acyclic graphs are used in many applications where path based analysis is performed. With algorithms targeting path analysis, the paths must not be partitioned for better performance per partition. We have presented a point handle based algorithm for graph partitioning. We have shown on some random graphs that the algorithm we have developed doesn't lead to a partitioning with paths getting partitioned.

Keyword: Depth First Search (DFS), Directed Acyclic Graph (DAG), point handles

I. INTRODUCTION

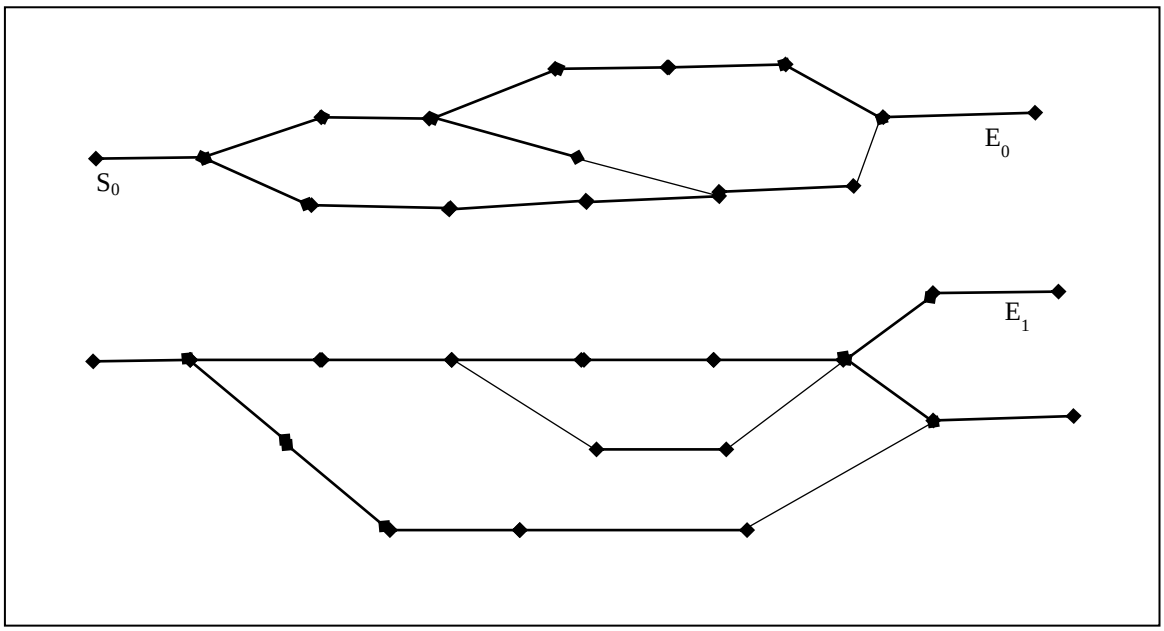

Fig.1 A sample Directed Acyclic Graph

Graph, in computer science is an abstract data type, which is used for representing the mathematical graph concept. A graph can be classified as directed or undirected, based upon the possibility of accessing the adjacent vertices from a given vertex in the graph. In directed graphs the vertices are connected with the edges which have direction associated with them, which means that all the adjacent vertices of a given vertex are not accessible with the same cost. Directed acyclic graphs or (DAG) is an important class of graph structures which,

is a finite directed graph with "no directed cycles". A DAG contains no vertex, v that can be reached to itself through a finite set of directed edges E [1]. Some of the important definitions, related to DAGs which we will be using throughout this paper are presented below:

1. Path – A path is a finite set of directed edges. Set of all paths is denoted by P.
2. Start Point – A start point is a vertex v, in the DAG which does not have any incoming edges to it. S is the set of all start points in the graph.
3. End Point – An end point is a vertex v, in the DAG which does not have any outgoing edges from it. E is a set of all end points in the graph.

Graph partitioning [11] is the process of splitting a graph into separate parts which can be used in different scientific and engineering problems where breaking a graph into smaller parts is needed, for example distributed graph analysis problems. In this paper we have proposed a new graph portioning algorithm for partitioning a DAG, in particular for applications which perform path based analysis.

Rest of the paper is organized as follows; we will describe briefly the point handle data structure in section 2. Section 3 describes the path based graph partitioning algorithms. Section 4 describes the comparative experiments and results, and conclusive remarks are made.

II. POINT HANDLE

Point handle (PH), is a vertex bound structure, which has a unique handle per vertex and it keeps the track of splitting and merging of paths from vertices in a DAG. Fig. 2 describes a PH structure in detail; where we have enlisted the main structure elements and define the basic intent of each one of them.

1. The source vertex handle, keeps the track of source vertex from where the PH is originated.
2. Master PH, keeps the track of PH from which the current PH is generated, it ensures that in constant time one can reach from a PH to its master PH, and can be reached to the origin point or a start point of the path.
3. Merged PH container contains a set of PHs which are merged into this PH and are getting propagated encapsulated in this PH.

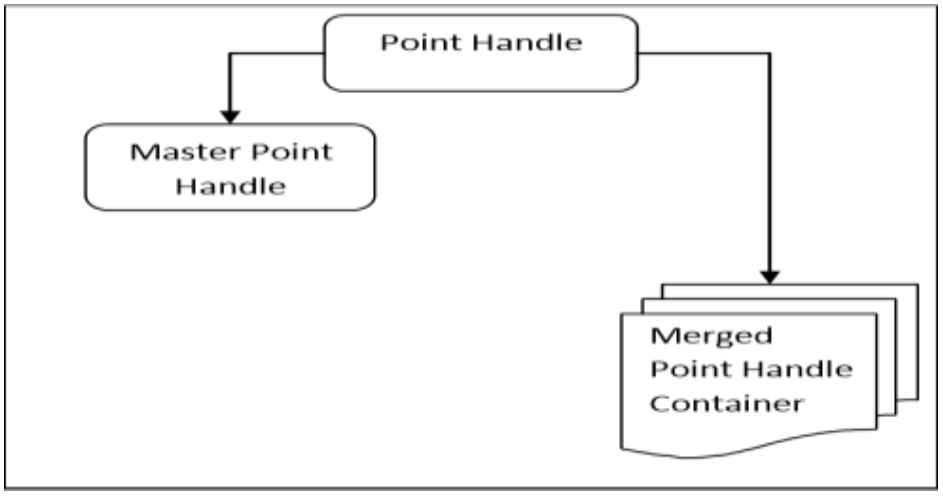

Fig.2 Point Handle Structure

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (02), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

A unique PH is created at each of the start points in the Graph and they are propagated on the paths originating from the vertices, but on each vertex of the graph we maintain only one PH explicitly. As described in Fig 3 the tag split and tag merge operations lead to a special structure called a tag graph. In Fig 4 point handle propagation through the example DAG is explained. A detailed account of PH propagation and associated algorithmic steps is presented in [10], where algorithm on path enumeration [2-9] is discussed.

III. GRAPH PARTITIONING USING POINT HANDLES

In this section we will present the graph partitioning algorithm using the point handles. One very important characteristic of the partitioning for path based analysis would be to keep the paths intact throughout the portioning process. For this purpose we have defined a special data structure called a Point Handle Tree.

3.1 Point Handle Tree

A point handle tree is an inverted tree structure which is rooted at a point handle at each end point in the DAG. Fig 5 shows the point handle graph of the example DAG of fig 1 and its resultant point handle tree.

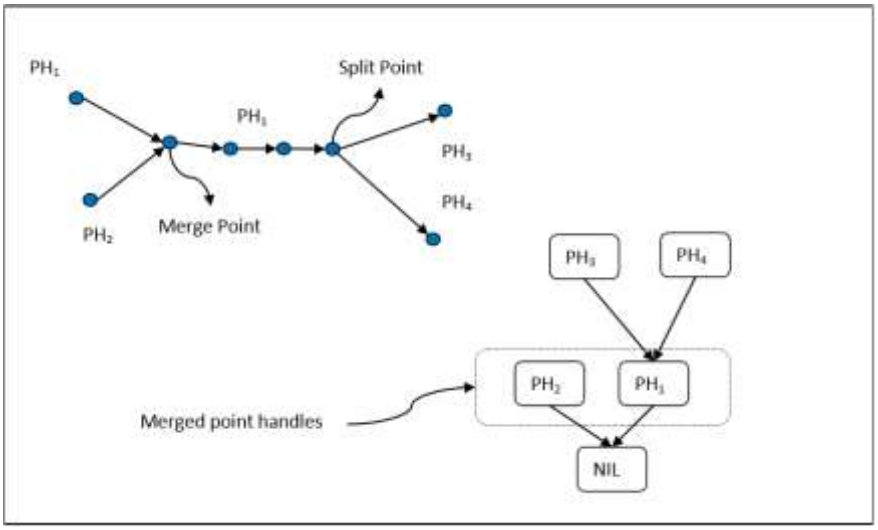

Fig.3 Merge and Split on the graph and point handle connectivity

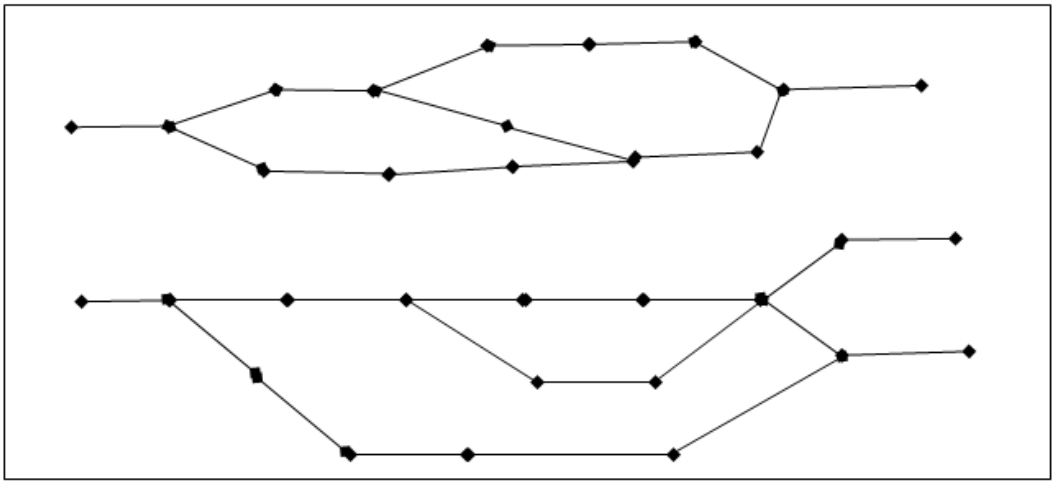

Fig.4 The point handle propagation on the DAG of Fig 1

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (02), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

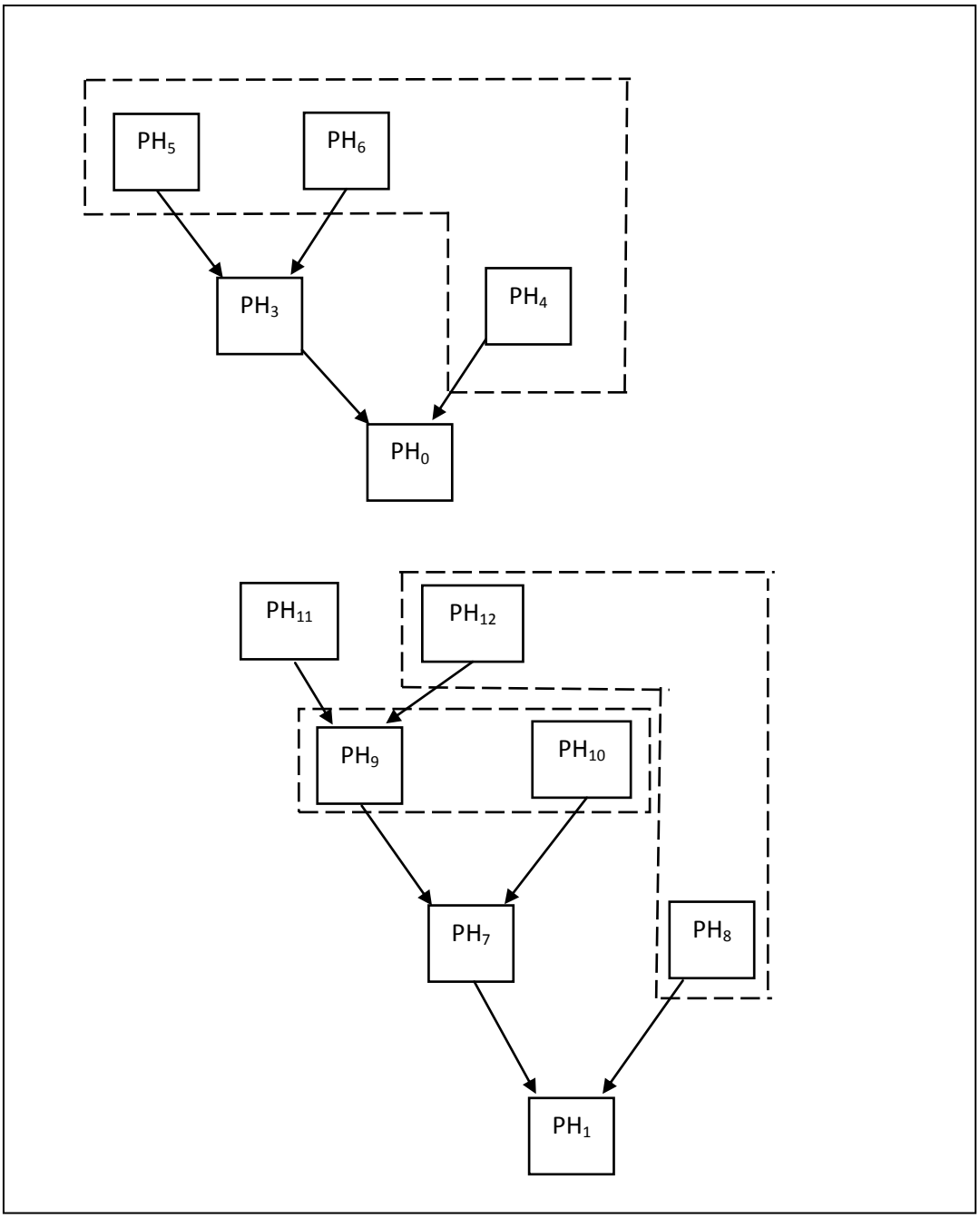

Fig 5 Point Handle Tree for the DAG in Fig 1

3.2 End Tag Unification

Once the point handle trees are populated for a given DAG, point handle unification is performed which is the process of marking the connected point handle trees. Uniquify_End_Point_Handle is the top level algorithm which is used for performing unification. Once unification is performed a unique ID is propagated to each of the point handles. This operation can be seen as coloring of the end point handles into unique colors, and hence each point in the original DAG gets colored to a unique color. This information is used for partitioning of the graph.

The algorithm for unification is presented below -

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (02), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

```
OPERATION:Enqueue_level_Point_Handle
    Input:Handle PH_T, and unique ID I_T
    Output:PH Queue, Q
        PH_Set ← Expand PH_Set (PH_T)
        foreach PH | PH Є PH_Set
                    if Uniqification[PH_T] ≠ ø
                            if Uniqification[PH_T] ≠ I_T
                        Union_PH_ID (Uniqification[PH_T], I_T)
                    continue
            Uniqification[PH] = I_T
            Q ← enqueue(Master[PH])
        end
```

```
ALGORITHM: Uniquify_End_Point_Handle
    Input: Directed Acyclic Graph
    Output: UniquePH Id set
        foreach end point e | e Є V
            I_E = get_unique_id
            Q ← enqueue (PH[e])
            while Q ≠ ø
                T = dequeue (Q)
                Enqueue_level_Point_Handle (T, I_E)
            end
        end
```

3.3 Graph Partitioning Using Connected End Point Handle Trees

Once the end point handles are uniquified, the partitioning can be performed as described in the partitioning algorithm, *Partition_DAG*. With the uniquification already in place, partitioning algorithm goes through each of the vertices in the DAG and put each vertex into its respective partition set. This is a fairly simple partitioning algorithm which will ensure that the paths are not partitioned. One of the very important applications of the partitioning algorithm is that the partitions which are created can easily be used for distributed path based analysis, and there will be no inter-processor communication because no paths will ever encompass two computing nodes in a distributed computing environment. This algorithm will also be able to break the graph into partitions equal to the number of connected end point handle trees.

```
OPERATION:Partition_DAG
        Input:  Graph T
        Output:  Graph Partitioned
            unique_set ← Uniquify_End_Point_Handles (T)
            for each vertex v | v Є V
                Partition[v] = unique_set_id[Tag[v]]
            end
```

IV. RESULT AND CONCLUSION

We have applied the graph partitioning algorithm on randomly created directed acyclic graphs. We have tried to partition the graphs into different number of partitions and computed the standard deviation in the sizes of the graphs for each partitioning run. The standard deviation in the sizes gives us the idea of quality of partitioning attained by performing the same on random graphs. In some cases, high number of partitions could not be created.

		Standard deviation in size of each partition			
Graph (#Nodes, #Edges)	**N=2**	**N=6**	**N=10**	**N=16**	**N=20**
1362, 1924	134	120	80	48	NA
958, 3088	94	68	56	NA	NA
2450, 2922	306	159	98	72	NA
1355, 1787	102	82	78	58	NA

As it can be seen from the results, there are two shortcomings of this algorithm –

1. It is not ultra-scalable, as it cannot lead to a large number of partitions and the number of partitions possible by this algorithm is equal to number of individually connected end point handle trees.

2. Partitions generated by this scheme are not very balanced in terms of number of vertices and paths per partition.

As a future extension of this work we may develop algorithms based on the idea of point handles which can lead to scalable and balanced partitions.

REFERENCES

[1] GIUSEPPE DI BATTISTA et al, DRAWING DIRECTED ACYCLIC GRAPHS: AN EXPERIMENTAL STUDY, Int. J. Comput. Geom. Appl. 10, 623 (2000). Reference of applications page where paths are needed

[2] Eppstein, 1998, D. Eppstein, Finding the *K* shortest paths, Journal of the Society for Industrial and Applied Mathematics, 28 (2) (1998), pp. 652-673

[3] Yau-Tsun Steven Li , Sharad Malik, Performance analysis of embedded software using implicit path enumeration, Proceedings of the 32nd annual ACM/IEEE Design Automation Conference, p.456-461, June 12-16, 1995, San Francisco, California, USA [doi>10.1145/217474.217570]

[4] Dial, R B, 1971 "A probabilistic multipath assignment model which obviates path enumeration" Transportation Research 5 (2) 83–111 Google Scholar, Crossref

[5] E. DeutschDyck path enumeration, Discrete Math., 204 (1999), pp. 167-202

[6] A. Marino, Analysis and Enumeration, Atlantis Studies in Computing 6, Atlantis Press and the authors 2015

[7] John J. Zasio, Kenneth C. Choy, Darrell R. Parham, "Static timing analysis of semiconductor digital circuits", US Parent US4924430 A, issued May 8, 1990

[8] Maria Gradinariu 1 Sébastien Tixeuil , "Self-stabilizing Vertex Coloring of Arbitrary Graphs", International conference on Principles of Distributed Systems (OPODIS 2000), Dec 2000, Paris, France. pp.55-70, 2000

[9] GuyMelançon, FabricePhilippe, "Generating connected acyclic digraphs uniformly at random", Information Processing Letters Elsevier, Vol 90 Issue 4, Pages 209-213, 2004

[10] S Singh and A K Saxena, "A Novel Path Enumeration Algorithm for Directed Acyclic Graphs", International Journal of Advanced Research in Science and Engineering, Vol 06 Issue 11, Pages 1852-1861

[11] G. Karypis, V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs", Technical Report 95-035, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1995.