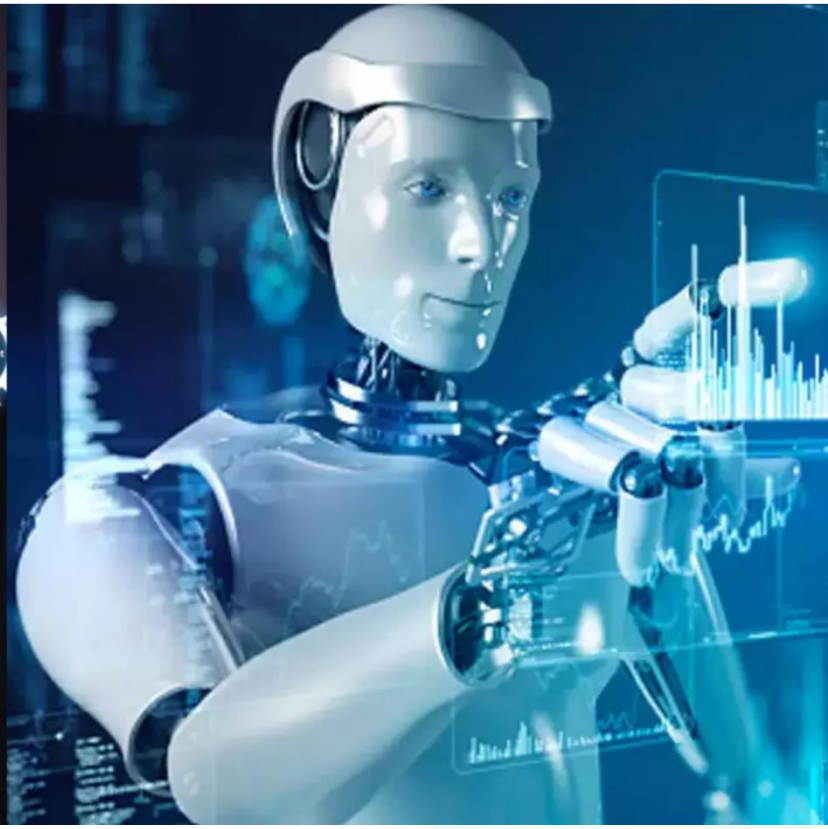
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 14, Issue 6, June 2025

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 6, June 2025

|DOI: 10.15680/IJIRSET.2025.1406165|

Tree Drawing Algorithm for Text Visualization using Computer Graphics

Dr.R.Maruthamuthu[1], Kandadi Lakshmi Deepak[2], Mopuri Bhavani[3], Pothala Poojitha[4], Phatan Naazeema[5]

Assistant Professor, Department of Master of Computer Applications, Madanapalle Institute of Technology and Science, Madanapalle, India[1]

PG Student Department of Master of Computer Applications, Madanapalle Institute of Technology and Science, Madanapalle, India[2,3,4,5]

ABSTRACT: Level-based tree drawing is a common algorithm that produces intuitive and clear presentations of hierarchi- cally structured information. However, new applications often introduces new aesthetic requirements that call for new tree drawing methods. In this paper, we propose an indented level- based tree drawing algorithm for visualizing parse trees of English language. This algorithm displays a tree with an aspect ratio that fits the aspect ratio of the newer computer displays, while presenting the words in a way that is easy to read. We discuss the design of the algorithm and its application in text visualization for linguistic analysis and language learning. An efficient and practical implementation of the algorithm is also presented.

I. INTRODUCTION

A tree drawing algorithm consists of a fixed of policies for putting the nodes and drawing the rims. some tree drawing rules are added to address the traits of the structure of the records. some tree drawing policies are aesthetic rules for the green use of space and readability of presentation. The maximum critical aesthetics of tree drawings include vicinity, factor ratio, subtree separation, closest leaf, and farthest leaf, and many others. a new application can also introduce new aesthetic policies that cause new tree drawing algorithms.we are developing a text analysis and visualization pro- gram for linguistic research and language getting to know. A parse tree is a rooted tree showing the syntactic shape of a sentence or a string. Visualizing the parse trees can assist researchers or college students examine the structure of the sentence and its complexity. the typical drawing of a parse tree is a top-down, stage-based totally tree (Fig. 1) but the disadvantage is that the factor ratio of the tree does not match the element ratio of more recent computer presentations. The tree grows vertically. the peak of the parse tree visualization is usually large than its width, mainly when the sentence is structurally sophisticated. With a dual monitor setup, the show is even wider. therefore a traditional parse tree visualization does no longer make most efficient use of the screen area. while more than one parse timber want to be displayed in a series, this trouble will become even extra apparent.

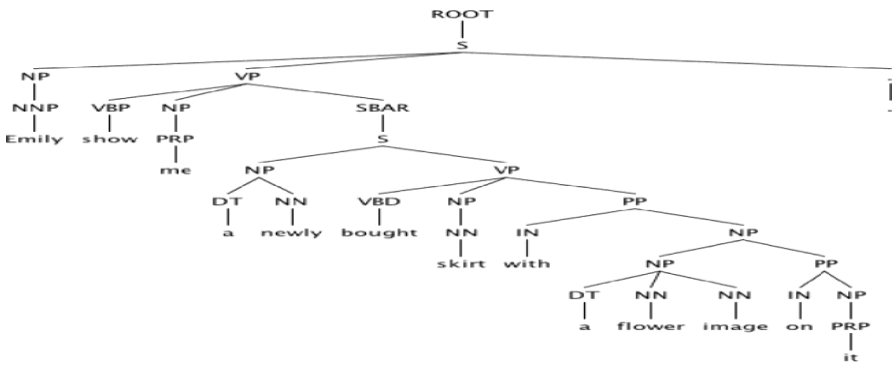


Fig. 1: The traditional level-based parsed tree of the sentence "Emily show me a newly bought skirt with a blue flower image on it

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 6, June 2025

|DOI: 10.15680/IJIRSET.2025.1406165|

II. RELATED WORK

Non-Layered Tree Drawing Algorithms:

The level-based tree drawing algorithms assume that nodes in the tree have uniform size. However, in many practical applications, these tree nodes may have varying sizes. Non- layered tree drawing algorithms are thereforedesigned to generate a more vertically compact drawing which places child nodes at a fixed distance from the parent nodes. Miyadera et al. present an O(n2) algorithm [8] for non- layered trees that horizontally positions parent nodes at a fixed offset from their first child, instead of centering above the children. Many other algorithms, such as Bloesch algorithm [9], Stein and Benteler algorithm [10] and Li and Huang algorithm [11], Marriot et al.

DEFINING INDENTED LEVEL-BASED TREE DRAWING PROBLEM:

The indented level-based tree drawing (see Fig. 2 for example) algorithm has the following properties:A left child and a right child should be positioned on the right of their parent node.The root node is the leftmost node of the tree. The tree is displayed horizontally.The horizontal coordinates of the leaf nodes should be indented in such a way that, starting with the top leaf node, the leaf nodes are sorted horizontally from left to right. the drawing is compact and satisfies the properties as listed above.

REINGOLD AND TILFORD ALGORITHM:

Reingold and Tilford Algorithm is one of the most impor- tant and influential algorithms in the area of tree drawings, and serves as the basis for our tree drawing algorithm. In this section, we discuss its basic ideas and the major challenges it has addressed.The brief description of the algorithm is as follows: two tree traversals are used to produce the final horizontal coordinates of nodes while their vertical coordinates can be pre-determined with their levels.The second pre-order traversal compute the final horizontal coordinates for each node by summing This process continues recursively until the root is reached.

INDENTED REINGOLD AND TILFORD TREE DRAWING ALGORITHM:

This property is in conflict with the Aesthetic rule #1 in the Reingold and Tilford algorithm. In our case, this is no longer valid. Therefore, our algorithm needs to compute both horizontal and vertical coordinates recursively.Here we still use a vertical tree for easier explanation.root relative to their children (if they have any) and This process is described in secondWalk proce- dure (See Algorithm 2). The last tree traversal redistributes the vertical positions of nodes in order to make it look more pleasant while still maintaining the properties of the indented tree drawing (See Algorithm 3).Like other positioning algorithms, our algorithm also uses two separate fields for the positioning of tree nodes. The position value in the modX or modY field of a node is assigned based on the entire subtree rooted at the node, and will be applied to all of its offspring nodes when calculating their final coordinates. For a leaf node, only the prelimX or prelimY is needed to denote its

Algorithm 1 firstWalk procedure:

```
1 : procedure FIRSTWALK(TreeNode v)
2:  TreeNode w            d v's left sibling
3:  List<TreeNode> nodesd child of v
 4:    int mid d center of child
5:int len → nodes.length
d Line 6-11 horizontal coordinates
6:if v is a non-leaf node then
7:mid=(nodes[0].prelimX+nodes[len- 1].prelimX)/2
                v.prelimX = mid;
9:else if w exists then
10:v.prelimX=w.prelimX+SSd SS: const
11:end if
d Line 12-19 vertical coordinates
12:if v is a non-leaf node then
13:v.prelimY=nodeY .prelimY+nodeY .modY- DIST;
d DIST: const
```

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 6, June 2025

|DOI: 10.15680/IJIRSET.2025.1406165|

d nodeY is v's child with the smallest vertical coordinates
14:**if** w exists **then**
15:v.modY=w.modY+DIST*w.l;
d w.l denotes the number of its offspring leaves
end if
17:**else if** w exists **then**
18:v.prelimY=w.prelimY+DIST;
19:**end if**
if w exists **then**
21:Positioning v on the right of w
22:**end if**
23: **end procedure**

Algorithm 2 secondWalk procedure:

1: **procedure** SECONDWALK(TreeNode v)
2:v.modX += v.parent.modX;
3:v.x = v.prelimX + v.modX;
d Line 2-3 Compute final horizontal coordinates
4:v.modY += v.parent.modY;
5:v.y = v.prelimY + v.modY;
d Line 4-5 Compute final vertical coordinates

8. END PROCEDURE

Relative horizontal or vertical position to its leftmost siblings. We assume in this paper that the coordinate system has its original point at the top-left corner.In order to decrease the complexity of computation,algorithm is designed to decouple the horizontal and vertical positioning of nodes. There are three major steps that will change the positions of the nodes. The first one is the initial assignment of both horizontal and vertical positions to nodes in accordance to the rules in the first tree traversal. The third step is the node redistribution step in the third tree traversal, which tunes the vertical positions of nodes. We discuss these three steps in more details in the coming sub-sections.

A.Assigning initial positions to nodes:

We have defined rules for the assignment of horizontal and vertical positions. For horizontal positioning, the rules state that (1) if a node is a non-leaf node, then place it in the center of its children; (2) if a node is(3) if a node is a leaf with a left sibling, then place it to the right of its left sibling at a pre- defined distance. The pseudo codes for these rules are listed in line 6-11 in Algorithm 1.The rules for assigning vertical positions to nodes are more complicated:

B.Positioning subtrees:

The horizontal positioning discussed above only calcu- lates the relative position of a node to its siblings or children. The task of positioning subtree aims at computing the relative position of the current subtree to its sibling subtree (Line 21 in Algorithm 1). Positioning a subtree on the right of its sibling subtree in the level-based tree drawing is a complicated process since the algorithm has to travel the sequence of right-most nodes in the left subtree and the sequence of leftmost nodes in the right subtree in order to determine the minimal shifting distance that can separate two subtrees at a pre-defined distance. (We have discussed this process in Section IV.) However, positioning subtrees in our tree drawing algorithm is greatly simplified and can be achieved in O(1) time.For a given subtree The proof is as follows: Assume the above statement is not correct. Since this is a non-leaf node, its offsprings contain at least one leaf node, and according to the second property of our tree drawing algorithm it is centered among its children. If the non-leaf node has only one leaf node, then it has the same horizontal coordinates as its leaf node.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 6, June 2025

|DOI: 10.15680/IJIRSET.2025.1406165|

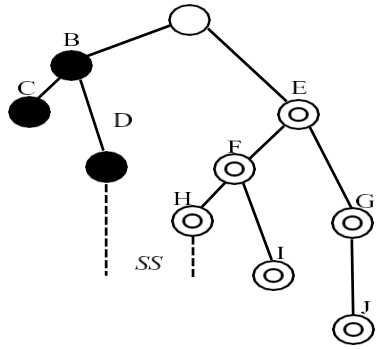


Fig. 3: Node D is the rightmost node in subtree B. Node H is the leftmost node in subtree E. The task of positioning subtree E is equivalent to separating Node D and Node H horizontally at a distance of SS (a pre-defined constant).

A. Redistributing nodes

In our algorithm, left sibling subtrees are gradually lifted up in order to line up the leaf nodes from left to right. In most cases this would not cause problems. subtree are not vertically distributed evenly. In Fig. 4, the tree drawing on the left does not look good as node A and node C is unnecessarily distant. The tree drawing on the right looks more pleasant after node C is repositioned between node A and node D.

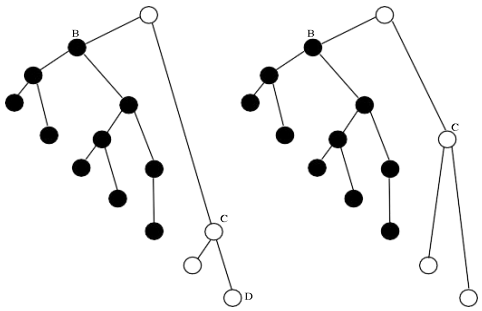


Fig. 4: The left tree drawing is less pleasant. The right tree drawing is more pleasant after node C is repositioned.

Algorithm 3 ThirdWalk procedure

1.procedure THIRDWALK(TreeNode v)
2:**for** each child u of v **do**
3:spots → (u.y-v.y)/DIST-1;
4:level → u.level
d for a leaf node, its level is 1
d for a non-leaf node, its level is 1 plus the max of its child's level
5:ave → spots/level
d ave: # of vertical positions u will be pulled toward its parent.
6:u.y → u.y-ave*DIST
7:end for
8: **E n d procedure**

To implement the node redistribution, the algorithm re- quires another pre-order tree traversal. The calculation is based on the idea that nodes at different levels within the visited node's subtree should be vertically and evenly positioned between the visited node and the leaf. As described in Algorithm 3, the thirdWalk procedure computes the allocatable space between the node and its child, evenly splitting the space among different levels of the subtree rooted on the child.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 6, June 2025

|DOI: 10.15680/IJIRSET.2025.1406165|

III. IMPLEMENTATION AND USAGE

We have implemented the indented level-based tree draw- ing algorithm in JavaScript and integrated it into D3.js [14], a popular JavaScript library for data visualization..The layouts package in D3.js library provides efficient implemen- tation of layout algorithms for various structures including the classic level-based tree. It also offers helper functions to facilitate the implementation of new layout algorithms.Based on D3.js, we have added new APIs for quick construction of the indented tree drawing.

APIs	Description
d3.layout.indentedtree	Creates a new indented tree layout.
indentedtree.size	Sets the available layout size.
indentedtree.sort	Sets the sort order of sibling nodes.
indentedtree.separation	Sets separation between nodes.
indentedtree.nodes	Computes the tree layout.
indentedtree.links	Returns edge positions.

Table I: APIs for the indented tree drawings

```
1
2          // Create a tree layout
3
4
5  var tree  = d3.layout.indentedtree();
6  var nodes = tree.nodes(root);
7                edges positions
8                tree.links(nodes);
9  // Retrieve
```

Listing 1: Sample code for drawing an indented tree,

Line 2 creates an instance of indented tree. Then in Line 3 the size of the tree drawing is specified. The positioning algorithm is invoked in Line 6, and tree nodes' horizontal and vertical coordinates are returned. The positions of tree edges are computed by pairing parent nodes and child nodes in Line 8. Once the position of tree nodes and edges are

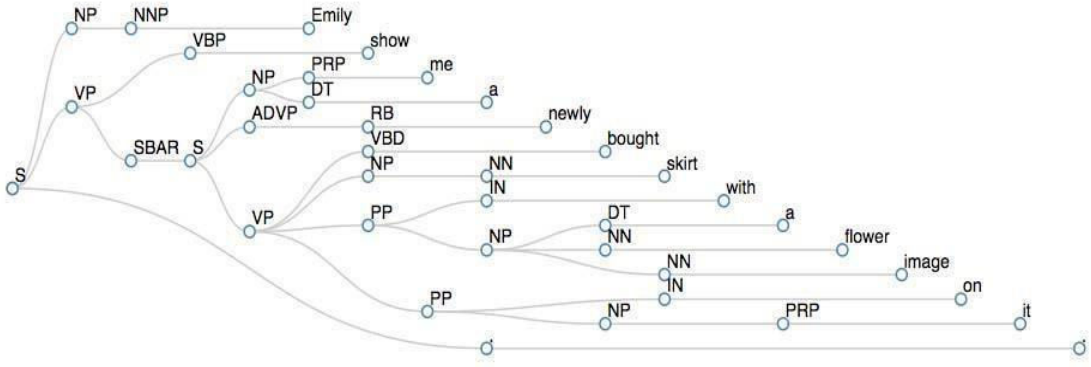


Fig. 5: The indented level-based parsed tree of the sentence "Emily show me a newly bought skirt with a blue flower image on it."

specified, then other D3 drawing routines can be used to draw the indented tree. Fig. 5 shows an example of the indentd level-based tree drawing created with our new APIs. We can see that our drawing has a wider aspect ratio that fits better with newer computer displays. It also places the words from left to right for easy reading, while clearly presenting the syntactic structure of the sentence.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 6, June 2025

|DOI: 10.15680/IJIRSET.2025.1406165|

IV. CONCLUSION

This is motivated by the need of visualizing parse trees in a text visualization application. The new algorithm creates a parse tree drawing that makes optimal use of the space while maintaining the word order for easy reading. The new algorithm also solves the problems of line crossings for tidy presentation.The proposed tree drawing algorithm is a useful comple- ment to the traditional level-based tree drawing algorithms. In addition to drawing parse trees, it can be applied to any tree structure where the leaf nodes need to be horizontally sorted.

REFERENCES

[1]C. Wetherell and A. Shannon, "Tidy drawings of trees," IEEE Transactions on Software Engineering, no. 5, pp. 514–520, 1979.
[2]R. Tamassia, "Handbook of graph drawing and visualization,"CRC Press 2013. [8]Y. Miyadera, K. Anzai, H. Unno, and T. Yaku, "Depth-first layout algorithm for trees," Information processing letters, vol. 66, no. 4, pp. 187–194, 1998.
[3]X. Li and J. Huang, "An improved generalized tree layout al- gorithm," in Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on, vol. 2. IEEE, 2010, pp. 163–166.
[4]K. Marriott, P. Sbarski, T. van Gelder, D. Prager, andA.
[5]M. Bostock, V. Ogievetsky, and J. Heer, "D3 data-driven doc- uments," IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 12, pp. 2301–2309, 2011.
[6] C. Buchheim, M. Ju¨nger, and S. Leipert, "Improving walkers algorithm to run in linear time," in Graph Drawing. Springer, 2002, pp. 344–353.
[7] R. Tamassia, "Handbook of graph drawing and visualization," CRC Press 2013.
[8] Y. Miyadera, K. Anzai, H. Unno, and T. Yaku, "Depth-first layout algorithm for trees," Information processing letters, vol. 66, no. 4, pp. 187–194, 1998.
[9] A. Bloesch, "Aesthetic layout of generalized trees," Software: Practice and Experience, vol. 23, no. 8, pp. 817–827, 1993.
[10] B. Stein and F. Benteler, "On the generalized box-drawing of treessurvey and new technology," in International Conference on Knowledge Management, 2007, pp. 416–423.
[11] X. Li and J. Huang, "An improved generalized tree layout algorithm," in Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on, vol. 2. IEEE, 2010, pp. 163–166.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details