Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 8, no. 3, pp. 295–312 (2004)

NP-Completeness of Minimal Width Unordered Tree Layout

Kim Marriott

School of Computer Science and Software Engineering
Monash University, Vic. 3800, Australia
marriott@mail.csse.monash.edu.au

Peter J. Stuckey

NICTA Victoria Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Vic. 3010, Australia
pjs@cs.mu.oz.au

Abstract

Tree layout has received considerable attention because of its practical importance. Arguably the most common drawing convention is the (ordered) layered tree convention for rooted trees in which the layout is required to preserve the relative order of a node's children. However, in some applications preserving the ordering of children is not important, and considerably more compact layout can be achieved if this requirement is dropped. Here we introduce the *unordered layered tree drawing convention* for binary rooted trees and show that determining a minimal width drawing for this convention is NP-complete.

Article Type	Communicated by	Submitted	Revised
Regular Paper	M. Kaufmann	May 2003	June 2005

Research supported in part by the Australian Research Council

1 Introduction

Rooted trees are widely used to represent simple hierarchies, such as organisation charts or family trees, and computer data structures. Because of their importance, layout of rooted trees has received considerable attention and many different algorithms have been developed for drawing rooted trees in a variety of different drawing conventions ranging from the standard layered tree convention to radial trees to hv-trees [2].

However, virtually all research has focused on drawing conventions in which the relative order of the children of a node is preserved in the layout. In many applications such as organisation charts the ordering of at least some of the children may not be important and for such "unordered trees" more compact tree layout may be obtained by varying the order of the children.

Here we investigate the complexity of tree layout for unordered binary trees. We show that determining a minimal width layout of an unordered rooted binary tree is NP-complete for a variant of the standard layered tree drawing convention. This partially answers an open problem in graph drawing (Problem 19 of [1]). Our proof of NP-hardness relies on a transformation from SAT [4].

This is in contrast to the case for ordered rooted binary trees for which a minimal width layered tree drawing can be found in polynomial time [5]. It is also in contrast to case for the hv-tree drawing convention where algorithms to compute drawings with minimal width and even minimal area layout for unordered binary trees have polynomial complexity ($O(n)$ and $O(n^2)$ respectively) [3]. But until now the complexity of unordered binary tree layout for the most common tree drawing convention, layered, has not been addressed.

In the next section we define exactly what we mean by minimal width layout of an unordered rooted binary tree and in Section 3 we show that the corresponding decision problem is NP-complete. Section 4 concludes.

2 The Problem Statement

We generally follow the terminology of [2]. A *binary tree* is a rooted tree in which each node may have up to two children. We represent binary trees by terms in the grammar

$$\text{Tree} \quad t \quad ::= \quad l \mid u(t) \mid b(t,t)$$

where l are leaf nodes, u are unary nodes, and b are binary nodes.

We are interested in the complexity of tree layout using the following *unordered layered tree drawing convention*. This requires that the drawing Γ of a binary tree (or a forest of binary trees):

- Is layered, that is the y-coordinate of a child is 1 less than the y-coordinate of its parent;
- Is planar, straight-line, and strictly downward;

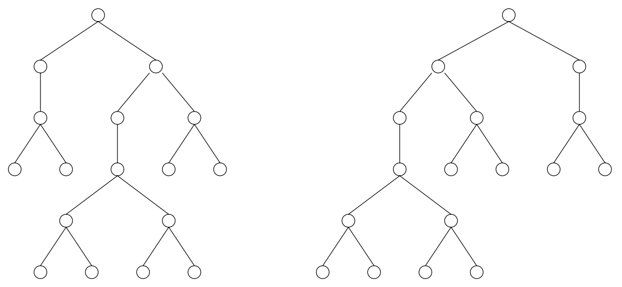


Figure 1: Two different unordered layered tree drawings of $b(u(b(l,l)), b(u(b(b(l,l), b(l,l))), b(l,l)))$.

- Any two vertices on the same layer are separated by at least 1;
- The x-coordinate of a parent is the average of the x-coordinates of its children.

Note that the last requirement means that if a node has a single child then the parent must be directly above the child. Also note that the layout does not need to preserve the relative order of the children of a node. However, the planarity condition means that for each binary node $b(t_1, t_2)$, the sub-tree t_1 will either be to the left of t_2 or to the right: the nodes in t_1 and t_2 cannot be interspersed.

Note that we do *not* require the x-coordinates take integral values (are placed on a grid), in this case even the ordered binary tree layout problem is NP-hard [5].

The *width* of a drawing Γ is the difference between the maximum x-coordinate and minimum x-coordinate occuring in the drawing.

For example, in Figure 1 we give two different drawings of the same binary tree which demonstrate that by reordering the children we can substantially reduce the width of the drawing.

We are interested in minimising the width of the drawing. Thus the corresponding decision problem is:

Unordered Layered Binary Tree Layout: Given a binary tree T and a real number W determine if there is an unordered layered tree drawing Γ for T which has width $\leq W$.

In the next section we show that this problem is NP-complete.

3 NP-completeness of Unordered Layered Binary Tree Layout

The unordered layered tree drawing convention is a variant of the usual layered tree drawing convention for binary trees except that it does not require the order of children to be preserved in the layout. More precisely, we define the *ordered* *layered tree drawing convention* to have the same requirements as the unordered layered tree drawing convention as well the requirement that the left-to-right ordering of the children is preserved in the layout.

Supowit and Reingold [5] have shown that the problem of finding the drawing of minimum width for a variant of the layered tree drawing convention for ordered trees can be done in polynomial time using linear programming.

A *configuration* for a binary tree T is an ordered binary tree O that corresponds to some fixed ordering of the children in the binary nodes in T. We will define configurations in terms of an equivalence relation $=_u$ between ordered binary trees, defined as follows

- $l =_u l$
- $u(O_1) =_u u(O_2)$ iff $O_1 =_u O_2$
- $b(O_1, O_2) =_u b(O_3, O_4)$ iff ($O_1 =_u O_3$ and $O_2 =_u O_4$) or ($O_1 =_u O_4$ and $O_2 =_u O_3$).

A *configuration* O for a binary tree T is any ordered binary tree for which $O =_u T$.

The configurations shown in Figure 1 are $b(u(b(l,l)), b(u(b(b(l,l), b(l,l))), b(l,l)))$ and $b(b(u(b(b(l,l), b(l,l))), b(l,l)), u(b(l,l)))$ respectively.

Lemma 1 *Binary tree T has an unordered layered tree drawing of width W iff there is a configuration O for T which has an ordered layered tree drawing of width W.*

Proof: By definition any layout Γ of a binary tree T defines a configuration O for T, hence Γ is a layout for the ordered tree O. □

Theorem 1 Unordered Layered Binary Tree Layout *is in NP.*

Proof: From Lemma 1 we simply guess a configuration O for T, and use linear programming to find the minimal width W' using the ordered layered tree drawing convention, then check that $W' \leq W$. □

We now prove that the problem is NP-hard. We do this by giving a reduction from SAT [4]. We assume a set of Boolean variables $x_1, \ldots, x_N$. A *literal* is one of x_i or $\neg x_i$. A *clause* D is is a disjunction $l_1 \vee \cdots \vee l_n$ of literals. A *problem* P is a set of clauses $\{D_1, \ldots, D_M\}$. The SAT problem for P is to find a valuation θ to variables $x_1, \ldots, x_N$ that satisfies each clause in P.

Our encoding makes use of "connector trees" to glue the various parts of the encoding together.

A *connector tree* C_n is defined as follows:

- C_1 is l,
- C_n is constructed from the complete binary tree of height $\lceil \log n \rceil - 1$ by replacing each leaf node in this tree by either a binary node $b(l,l)$ or a unary node $u(l)$ such that the total number of leaf nodes is n.

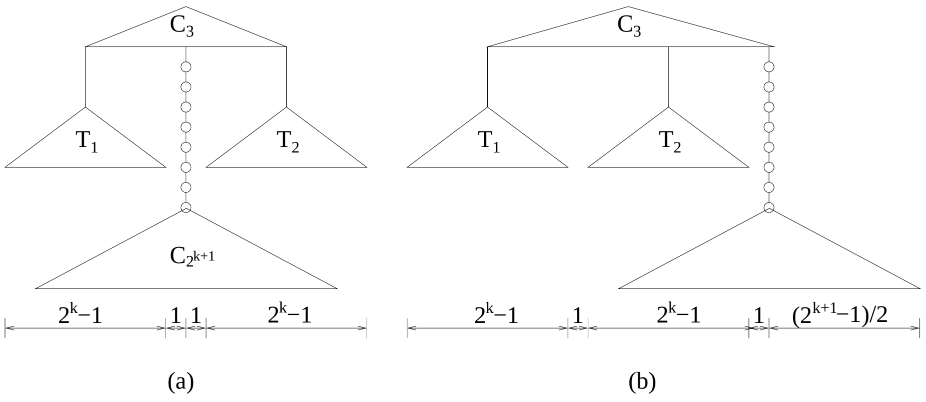


Figure 2: Two possible configurations of a tree with a "separator" sub-tree.

For example, C_3 is $b(b(l,l),u(l))$.
It follows from the construction that

- C_n has n leaf nodes all at the bottom of the tree
- C_n has height $\lceil \log n \rceil$
- C_{2^k} is the complete binary tree of height k.

Constructor trees allow us to glue other trees together without increasing the width of the layout:

Lemma 2 *Let the binary tree forest $T_1,\ldots,T_n$ have an (ordered) layered tree drawing of width W in which all root nodes are on the same level. Then the tree $T[T_1,\ldots,T_n]$ obtained from C_n by replacing the i^{th} leaf node by the unary node $u(T_i)$ also has a (ordered) layered tree drawing of width W.*

Our encoding of SAT relies crucially on two special types of trees: full binary trees C_{2^k} and vertical bars of the form $u(u(u(\ldots)))$. For each of these trees each configuration is isomorphic, allowing us to reason about all possible configurations of trees made up of these simpler elements reasonably straightforwardly. In particular, our encoding relies upon using the following construction to ensure that certain subtrees are forced to be next to each other in a layout of less than a fixed width.

Observation 2 *Let T be the binary tree shown in Figure 2 in two possible configurations. The subtrees T_1 and T_2 are required to each have 2^k nodes at some level L. In any unordered tree layout drawing of T with width less than $3\cdot 2^k - 1/2$ the $C_{2^{k+1}}$ subtree will separate T_1 and T_2. I.e. T must have configuration (a) or its mirror image.*

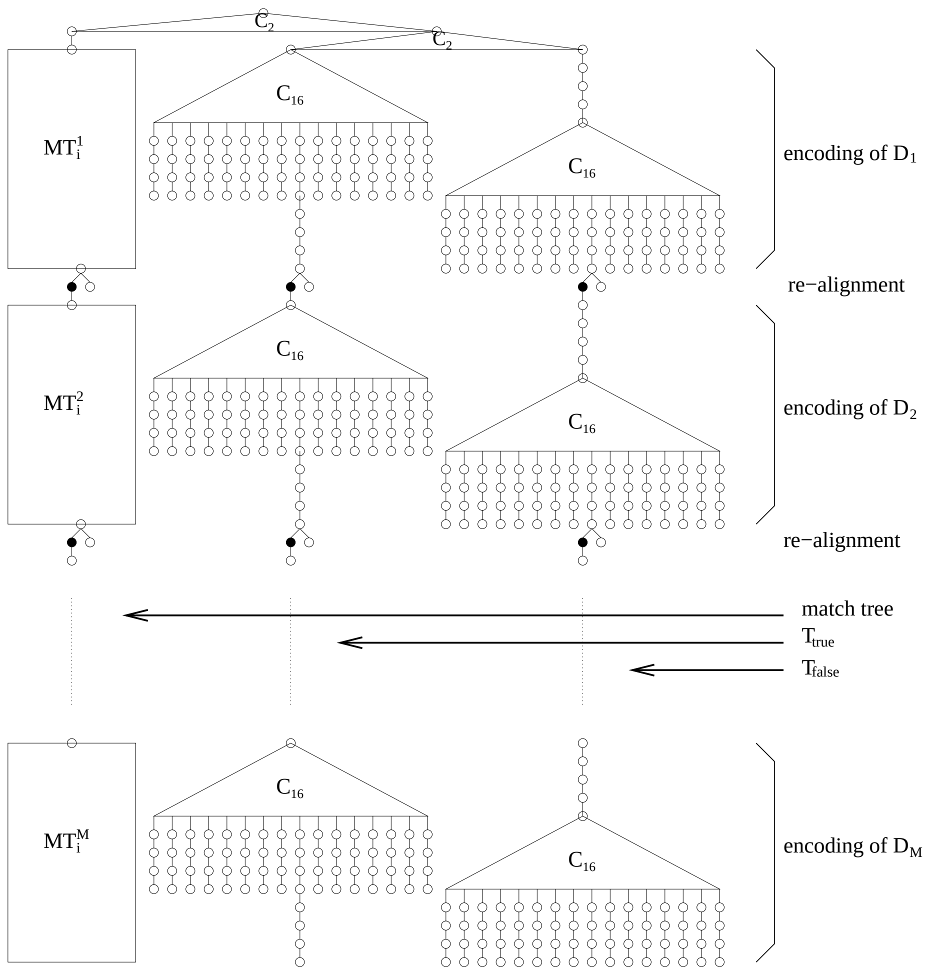


Figure 3: Tree T_{x_i} corresponding to variable x_i in the encoding of SAT.

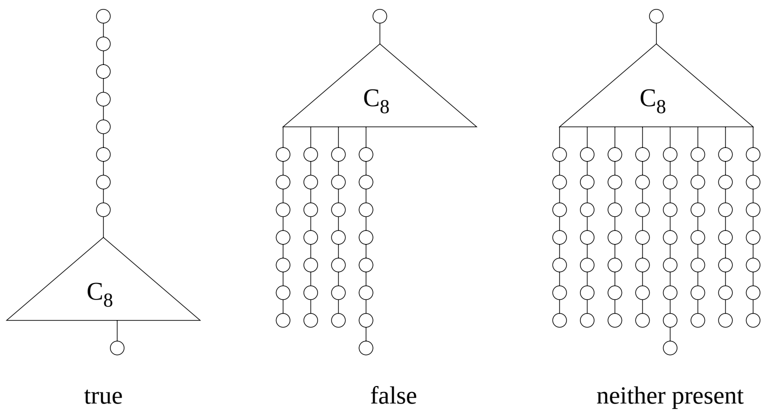


Figure 4: Subtrees corresponding to (a) x_i occurs in the clause, (b) $\neg x_i$ occurs in the clause, and (c) neither x_i nor $\neg x_i$ occur in the clause.

Proof: If T_1 and T_2 are adjacent, for instance as shown in configuration (b), then the width must be at least $(2^{k+1} - 1)/2 + 2^k + 2^k = 3 \cdot 2^k - 1/2$. $\square$

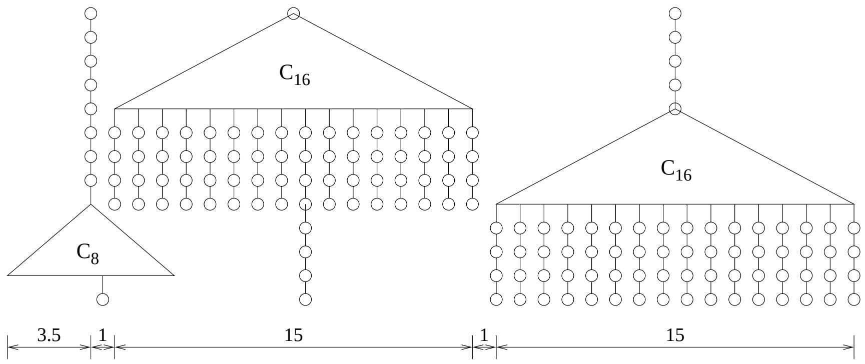
We assume that the SAT problem instance P being encoded has clauses $D_1, \ldots, D_M$ over variables $x_1, \ldots, x_N$. For simplicity we assume that N is 2^K for some K (we can always add up to N additional dummy variables to make this so.)

Corresponding to each x_i we construct the tree T_{x_i} shown in Figure 3. This has 3 main subtrees, in this configuration we have, from left to right, the "clause match" subtree, the subtree T_{true} corresponding to the assignment *true* and the subtree T_{false} corresponding to the assignment $false$ for this variable.

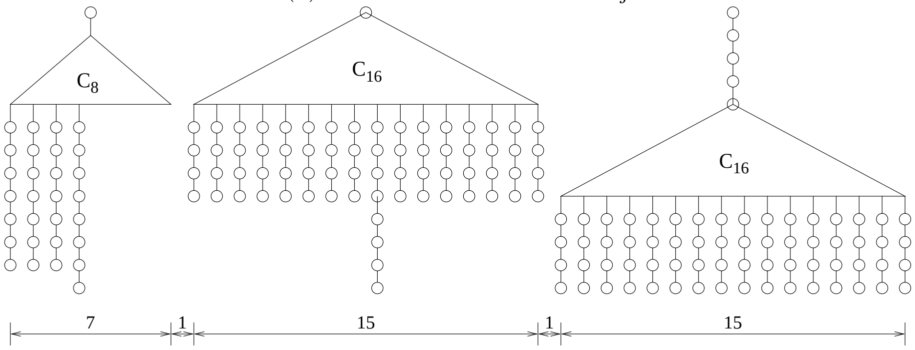
These trees have a repeated structure for each clause, so the tree T_{x_i} has M layers, one for each clause. In each level only the structure of the clause matching subtree changes. The three possible subtrees in the clause match subtree are shown in Figure 4. Between each layer there is a "re-alignment" level of C_2 subtrees. This allows the different layers to move relative to each other.

Now consider any configuration for T_{x_i}. From the construction of the tree either T_{false} or T_{true} must be adjacent to the clause matching subtree. This corresponds, respectively, to assigning x_i the value $false$ or $true$.

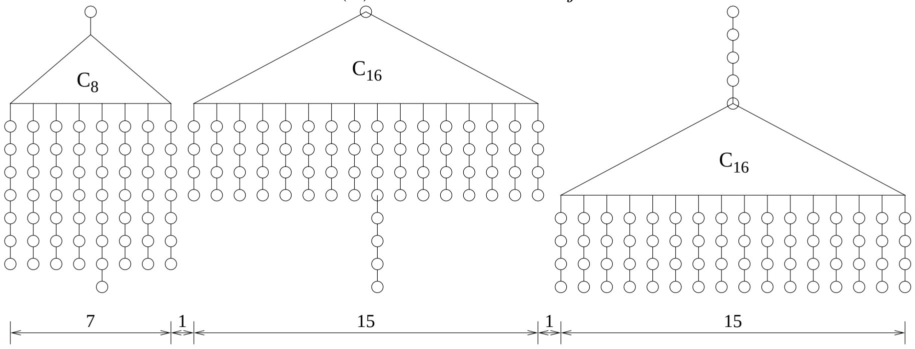
Now consider the minimum width of each clause level in the tree for each of these configurations. The different cases for the configuration in which T_{true} is adjacent to the clause matching subtree are illustrated in Figure 5. Figure 5 (a) shows the case in which there is a match, i.e. the clause matching subtree encodes that x_i occurs in the clause. It is obvious from the figure that the minimum width of the tree at that level is 35.5. Figure 5 (b) and (c) respectively show the cases in which the clause matching subtree encodes that $\neg x_i$ occurs in the clause or that neither x_i nor $\neg x_i$ occur in the tree. Again it should be clear



(a) x_i occurs in the clause D_j

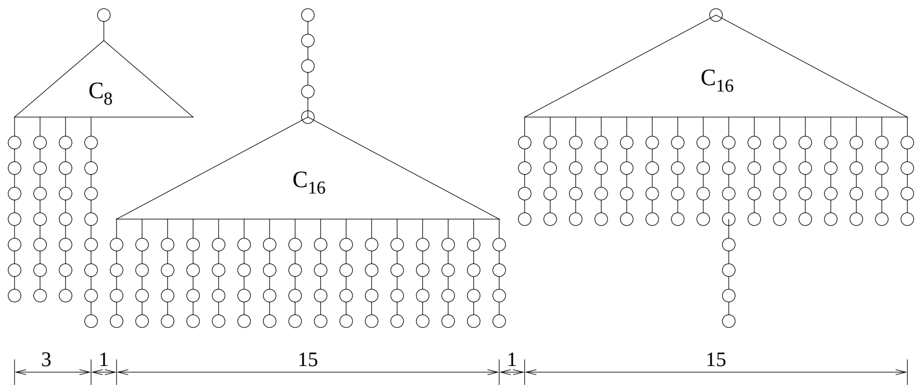


(b) $\neg x_i$ occurs in D_j

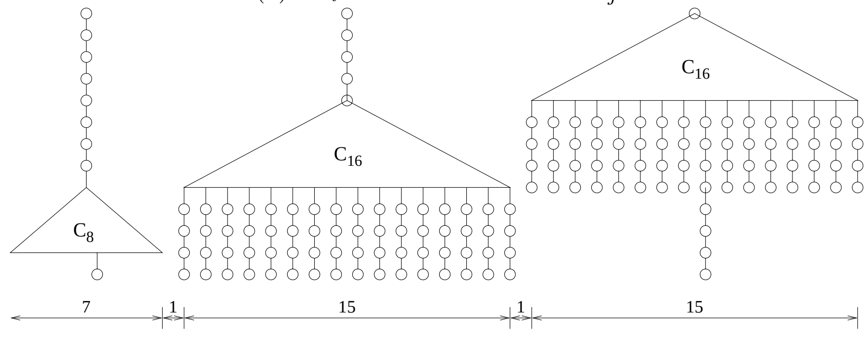


(c) neither $\neg x_i$ nor x_i occur in D_j

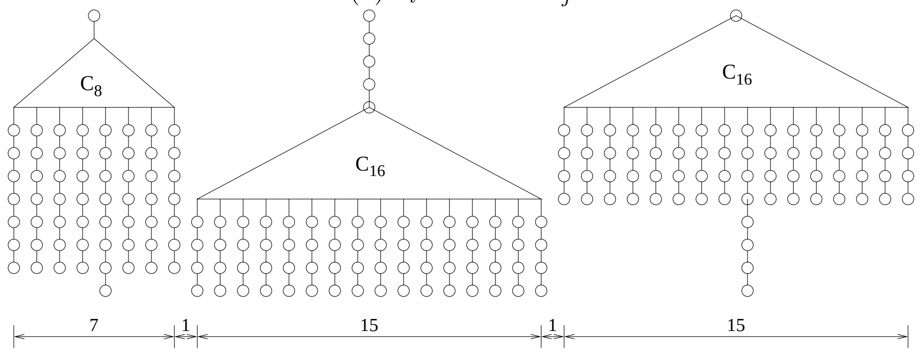
Figure 5: Minimum width of T_{x_i} at level j for the case corresponding to assigning *true* to x_i.



(a) $\neg x_i$ occurs in the clause D_j



(b) x_i occurs in D_j



(c) neither $\neg x_i$ nor x_i occur in D_j

Figure 6: Minimum width of T_{x_i} at level j for the case corresponding to assigning $false$ to x_i.

that for both cases the minimum width of the tree at that level is 39.

The different cases for the configuration in which T_{false} is adjacent to the clause matching subtree are illustrated in Figure 6. Figure 6 (a) shows the case in which there is a match, i.e. the clause matching subtree encodes that $\neg x_i$ occurs in the clause. In this case the minimum width of the tree at that level is 35. The remaining two cases are shown in Figures 6 (b) and 6 (c). For both cases the minimum width of the tree at that level is 39.

Note that if the order of children of the topmost tree C_2 in tree T_{x_i} is reversed, the match tree is to the right of T_{true} and T_{false} and the minimal width configurations for each level are simply mirror images of those shown in Figure 5 and Figure 6.

Thus, we have:

Observation 3 *If a drawing of* T_{x_i} *has width less than 39 at the level encoding clause* D_j*, the assignment to* x_i *satisfies* D_j*.*

The final step in the encoding is to combine the subtrees $T_{x_1}, \ldots, T_{x_N}$ for each variable into a single tree. The trick is to ensure that the assignments to different variables cannot interfere with each other in a layout of minimal width. We use a separator subtree to ensure this.

We define the tree $T_k(y_1, \ldots, y_{2^k})$ for encoding the variables $y_1, \ldots, y_{2^k}$ recursively as follows. The tree $T_0(y_1)$ is simply T_{x_i} where y_1 is x_i. The tree $T_{k+1}(y_1, \ldots, y_{2^{k+1}})$ is constructed from $T_k(y_1, \ldots, y_{2^k})$ and $T_k(y_{2^k+1}, \ldots, y_{2^{k+1}})$ by placing a separator subtree between them as shown in Figure 7.

Notice how $T_{k+1}(y_1, \ldots, y_{2^{k+1}})$ has C_3 trees at each re-alignment level to allow the layers to move relative to each other.

The tree $T_K(x_1, \ldots, x_N)$ where $N = 2^K$ is the complete encoding of our instance of SAT. For example, Figure 8 shows the encoding of $(A \vee B) \wedge (\neg B)$.

We say that a configuration or drawing for $T_k(y_1, \ldots, y_{2^k})$ is *valid* if there is a separator tree between each T_{y_i} in the tree.

Lemma 3 *Any tree* $T_K(x_1, \ldots, x_N)$ *has an unordered layered tree drawing of width* $41 \cdot (N-1) + 39$ *or less.*

Proof: This follows from the construction. Choose any valid configuration O for $T_K(x_1, \ldots, x_N)$. Consider each clause D_j. In each T_{x_i}, the layer corresponding to D_j can be drawn in width 39. If we consider this layer in isolation to the rest of the tree, taking into account the vertical bar of the separator subtree between adjacent T_{x_i}'s, the level can be drawn in width $39 \cdot N + 2 \cdot (N-1) = 41 \cdot (N-1) + 39$. Now by construction the C_2 and C_3 trees in the re-alignment level between each level can be used to exactly align the layers. The bottom parts of the tree (corresponding to the bottoms of the separation subtrees) can be drawn in width less than this, and from Lemma 2 the connectors at the top of the tree do not increase the width of the tree. □

The role of the realignment level is to allow us to shift layers relative to each other. This allows us to draw trees corresponding to satisfiable problems

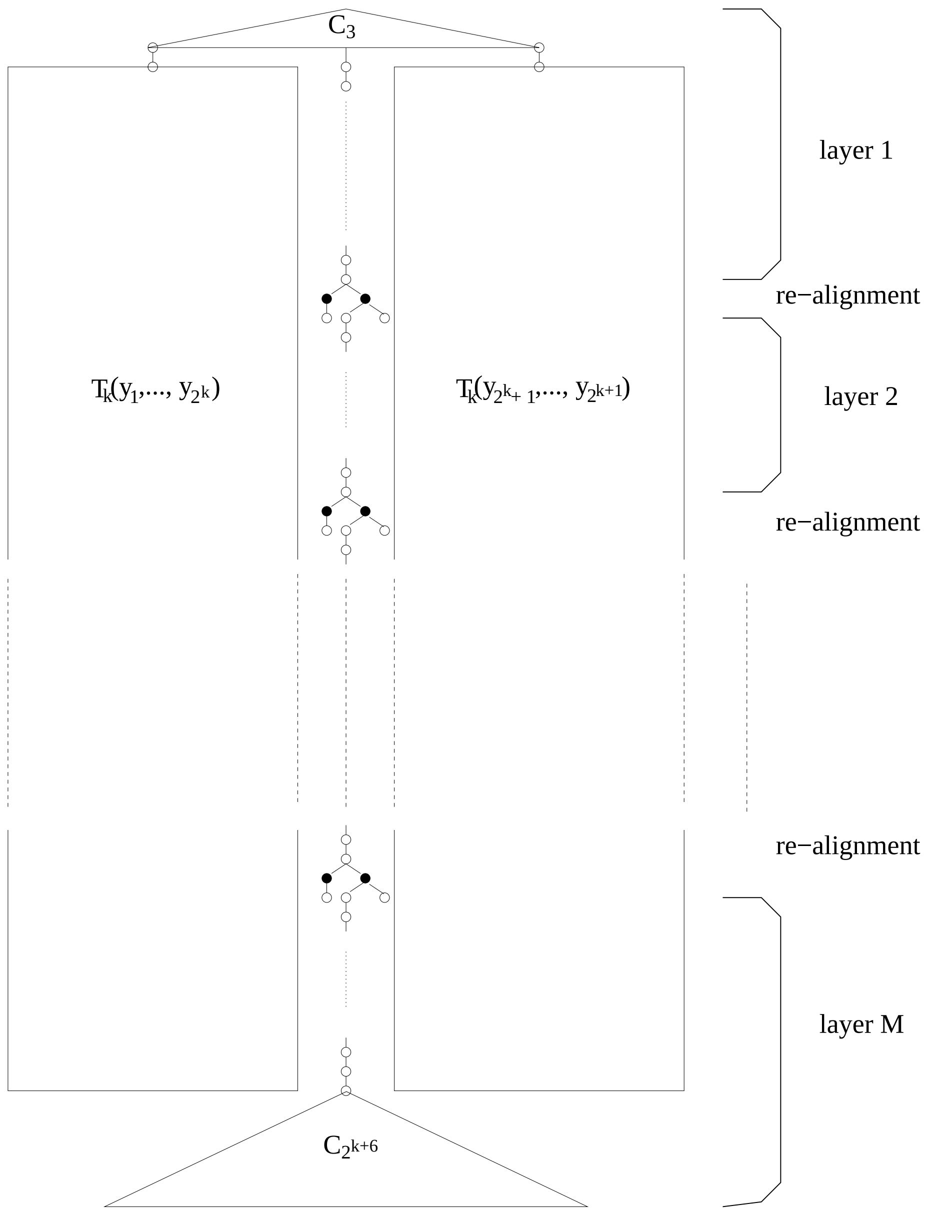


Figure 7: The tree $T_{k+1}(y_1, \ldots, y_{2^{k+1}})$

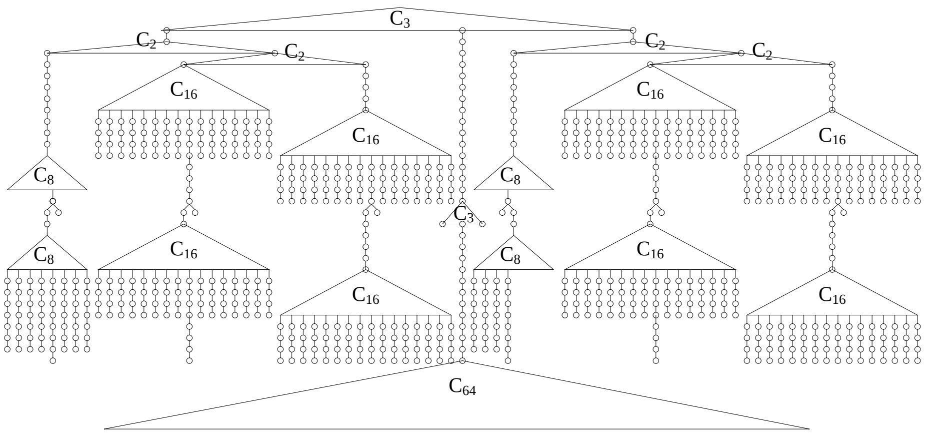


Figure 8: The tree encoding the SAT problem $(A \vee B) \wedge (\neg B)$.

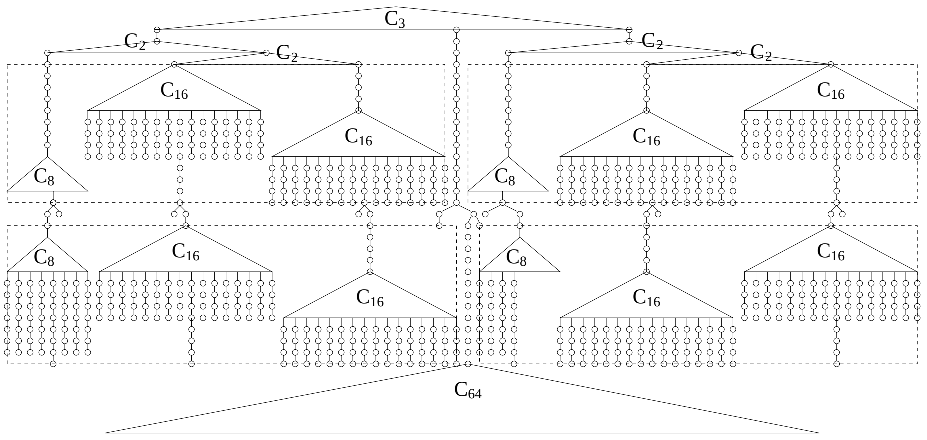


Figure 9: Block view of the configuration shown in Figure 8, illustrating width 79 construction.

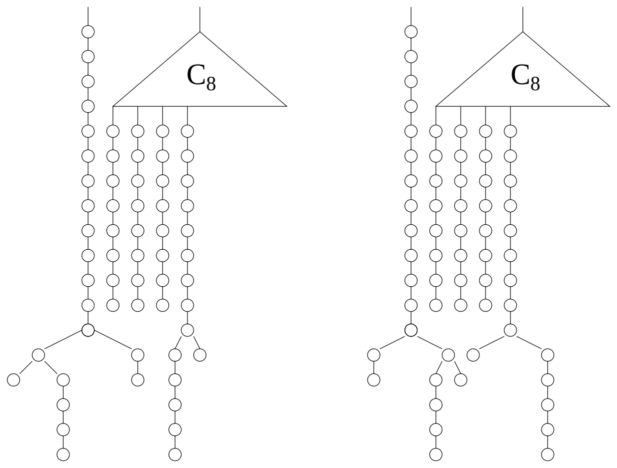


Figure 10: Realigning the tree levels one to the left, or one to the right, for the case when a separator subtree is adjacent to a clause matching subtree.

in narrower width trees. For example, Figure 9 shows a drawing of the minimal width configuration for the tree in Figure 8. This configuration is valid and corresponds to the assignment $A = true$ and $B = false$ which is a solution of the original problem. The four dashed outlines illustrate the four blocks of interest. The top leftmost block has width 38, indicating that $A = true$ satisfies the clause $A \vee B$ at the first literal. The bottom rightmost block has width 38 indicating $B = false$ satisfies the clause $\neg B$ in this literal. The other blocks have width 39 indicating the clauses are not satisfied at this point. To accommodate a overall minimal width of $79 = 38+2+39$ there is a shift right of one in the trees T_{true} and T_{false} on the left, at the C_3 tree in the re-alignment level and in the T_B tree.

Lemma 4 *If $T_K(x_1, \ldots, x_N)$ encodes a SAT problem which is satisfiable then it has an unordered tree layout drawing of width $41 \cdot (N-1) + 38$ or less.*

Proof: Let θ be a solution of the SAT problem. Choose a valid configuration O for $T_K(x_1, \ldots, x_N)$ that corresponds to θ in the sense that the configuration for each T_{x_i} corresponds to assignment in θ for x_i.

We slightly modify the construction given in the proof of Lemma 3. Consider each clause D_j. Since θ is a solution, for some x_i either x_i occurs in D_j and $\theta(x_i) = true$ or $\neg x_i$ occurs in D_j and $\theta(x_i) = false$. It follows that T_{x_i} at the level corresponding to D_j can be drawn in width 38. In all other $T_{x_{i'}}$, $i' \neq i$, the layer for D_j can be drawn in width 39. Thus if we consider this layer in isolation to the rest of the tree, taking into account the enforced separation between adjacent T_{x_i}'s, the level can be drawn in width $41 \cdot (N-1) + 38$.

Now because of the re-alignment level between each layer, the different layers are free to move by 1 relative to each other. To see that this is possible consider

Figure 9. The re-alignment level consists of C_2 and C_3 trees. In most cases their roots are separated by at least 8, allowing them to expand without interference. The only point where they may be closer is when a separator subtree is adjacent to a clause matching subtree in which case we have a C_3 tree whose root is only 4 apart from that of a C_2 tree. Figure 10 shows the layout for this case and that we can still shift by 1 to the left or to the right.

This means that we can draw each layer in width $41 \cdot (N-1) + 38$ or less. As before the bottom parts of the tree can be drawn in width less than this and the connectors at the top of the tree do not increase the width of the tree. □

We now prove the converse of this lemma: that if $T_K(x_1, \ldots, x_N)$ encodes a SAT problem which is unsatisfiable then it does not have a drawing of width $41 \cdot (N-1) + 38$ or less. In order to do so we actually show that it does not have a drawing of width $41 \cdot (N-1) + 38$ or less using a drawing convention which relaxes the unordered tree layout drawing convention so as to allow the layers in the tree to be treated independently. This allows us to reason recursively about minimal width layouts.

More precisely, we define the *layer independent drawing convention* for a tree $T_k(y_1, \ldots, y_n)$ to be the same as the ordered layered tree convention except that the solid black nodes shown in the re-alignment levels in Figure 3 and Figure 7 are not constrained to have their x-coordinate at the average of the x-coordinate of their children. This means that in a layer independent drawing of $T_k(y_1, \ldots, y_n)$ we can move the layers relative to each other and so we can minimise the width of each layer independently of the other layers.

Let Γ_O be a layer independent drawing for some configuration O of $T_k(y_1, \ldots, y_n)$ encoding clauses $D_1, \ldots, D_M$. We let $width_j(\Gamma_O)$ denote the width of the drawing of the j^{th} layer. As shown in Figure 7 the j^{th} layer consists of those parts of the tree encoding clause D_j and, in the case of layer 1 it also includes the connectors at the top of O and in the case of layer M the base of O. Since the width of each layer is independent it is possible to find a minimal ordered tree layout for a particular configuration which minimises all of these widths. We call such a drawing *layer minimal*.

Let O_1 and O_2 be configurations for some $T_k(y_1, \ldots, y_n)$ and let Γ_{O_1} and Γ_{O_2} be layer independent drawings for O_1 and O_2 respectively which are layer minimal. We say that O_1 *better minimises layer width* than O_2 if
(a) for all $1 \leq j \leq M$, $width_j(\Gamma_{O_1}) \leq width_j(\Gamma_{O_2})$, and
(b) for some $1 \leq j' \leq M$, $width_{j'}(\Gamma_{O_1}) < width_{j'}(\Gamma_{O_2})$.
The configuration O for some $T_k(y_1, \ldots, y_n)$ has *minimal layer width* if there is no other configuration for $T_k(y_1, \ldots, y_n)$ that better minimises layer width.

It follows immediately from the proof of Lemma 3 that

Lemma 5 *If O is a valid configuration of some $T_k(y_1, \ldots, y_n)$ and Γ_O is a layer minimal drawing for O then for all $1 \leq j \leq M$, $width_j(\Gamma_O) \leq 41 \cdot (n-1) + 39$.*

Lemma 6 *If O is a minimal layer width configuration for $T_k(y_1, \ldots, y_n)$ then O is a valid configuration.*

Proof: The proof is by induction on k. If $k = 0$ then it is trivially true since any configuration for tree $T_0(y_1)$ is valid.

Now consider any minimal layer width configuration O for $T_{k+1}(y_1, \ldots, y_n)$ (where $n = 2^{k+1}$). From Lemma 5, we know that $width_M(\Gamma_O) \leq 41 \cdot (n-1) + 39$ where Γ_O is a layer minimal drawing for O. It follows from the definition that T_{k+1} has the tree $C_{2^{k+6}}$ at its base and that each of the T_k sub-trees effectively has the tree $C_{2^{k+5}}$ at its base. Thus, from Observation 2, if the M^{th} layer in O has a drawing with width less than $3 \cdot 2^{k+5} - 1/2$, the "separator" subtree in T_{k+1} must separate the two T_k subtrees. Now

$$41 \cdot (2^{k+1} - 1) + 39 = 82 \cdot 2^k - 2 < 96 \cdot 2^k - 1/2 = 3 \cdot 2^{k+5} - 1/2$$

for all $k \geq 0$ so in any minimal layer width configuration, and thus for O, the separator tree must separate the two T_k subtrees.

But this means that for each layer $1 \leq j \leq M$, the vertical separator "bar" must separate the two T_k subtrees. Thus the minimal width of O on each layer is simply 2 plus the minimal width of the T_k subtrees on that layer. Hence, since O is a minimal layer width configuration for T_{k+1} it must also be a minimal layer width configuration for the two T_k subtrees. By induction, therefore O is a valid configuration for the two T_k subtrees. Therefore O is a valid configuration since from above, the separator tree in O must separate the two T_k subtrees. □

Lemma 7 *Let $T_K(x_1, \ldots, x_N)$ encode a SAT problem which is unsatisfiable. The minimal width of any layer independent drawing of some configuration of $T_K(x_1, \ldots, x_N)$ is $41 \cdot (N-1) + 39$.*

Proof: Consider some minimal width layer independent drawing Γ_O for $T_K(x_1, \ldots, x_N)$ where O is the corresponding configuration.

W.l.o.g. we can assume that O is a minimal layer width configuration. Thus from Lemma 6, O is a valid configuration. From Lemma 5, for all $1 \leq j \leq M$, $width_j(\Gamma_O) \leq 41 \cdot (N-1) + 39$. Now consider θ the valuation corresponding to the truth assignment in O. Since the SAT problem is unsatisfiable, θ is not a solution for some clause $D_{j'}$. From the definition of O it follows that $width_{j'}(\Gamma_O) = 41 \cdot (N-1) + 39$. Thus the width of Γ_O is $41 \cdot (N-1) + 39$. □

Corollary 4 *Let $T_K(x_1, \ldots, x_N)$ encode a SAT problem which is unsatisfiable. The minimal width of an unordered layered tree drawing of $T_K(x_1, \ldots, x_N)$ is greater than or equal to $41 \cdot (N-1) + 39$.*

Proof: Since the layer independent drawing convention is less restrictive than the ordered layered tree drawing convention, it follows from Lemma 7 that the minimal width of any ordered layered tree drawing of some configuration of $T_K(x_1, \ldots, x_N)$ is greater than or equal to $41 \cdot (N-1) + 39$. The result follows from Lemma 1. □

Putting Lemma 4 and Corollary 4 together we have

Proposition 5 *Let $T_K(x_1, \ldots, x_N)$ encode an instance P of SAT. $T_K(x_1, \ldots, x_N)$ has a drawing of width less than $41 \cdot (N-1) + 39$ iff P is satisfiable.*

Now consider the size of the tree $T_K(x_1, \ldots, x_N)$ encoding an instance P of SAT with N variables and and M clauses. $T_K(x_1, \ldots, x_N)$ has $O(N)$ nodes on each level and the height of the middle part of the tree encoding the clauses is $O(M)$, the height of the top of the tree is $O(\log N)$ and the height of bottom of the tree is $O((\log N)^2)$ since the height of the complete tree at the bottom of each separator node is $O(\log N)$ and there are $O(\log N)$ tiers of these complete trees. Thus, the total height is $O(M + (\log N)^2)$ and so the encoding has polynomial size and can clearly be generated in polynomial time. Hence, from Proposition 5, we have the following.

Theorem 6 UNORDERED LAYERED BINARY TREE LAYOUT *is NP-hard.*

Therefore from Theorem 1:

Corollary 7 UNORDERED LAYERED BINARY TREE LAYOUT *is NP-complete.*

4 Conclusion

Despite the practical importance of layered tree layout and unordered trees, the complexity of layered tree layout for unordered trees was not known. Here we have shown that for the case of binary trees it is NP-complete. While it is clear that naive algorithms for minimal width unordered layered binary tree layout have exponential complexity, since the obvious approach is to try different possible configurations of the tree, proving NP-completeness was surprisingly difficult. Basically the difficulty arises because most unordered trees have many different configurations and so are difficult to reason about. Our construction therefore relies on using two types of binary tress for which all configurations are isomorphic: complete binary trees and vertical "bars" made from unary trees.

Our result naturally generalises to n-ary trees and to trees in which there is a partial ordering on children in a node, i.e. only the relative ordering between some children needs to be preserved in the layout.

However the proof depends crucially on the drawing convention that a node's x coordinate is the average of its children's x coordinates. Indeed, if this requirement is removed from the drawing convention then it is easy to determine the minimum width drawing in linear time: it is simply the maximum number of nodes occurring on any layer in the tree. One obvious question is what happens if we change the drawing convention so that for nodes with a single child there is no need for the node to be directly above the child. We believe the problem is still NP-complete, but it is not obvious how to modify the construction given here to prove this since we can no longer argue that unary trees form a vertical bar.

Acknowledgements

We thank Franz Brandenburg who identified the complexity of layered tree layout for unordered trees as an open problem in graph drawing in his lectures on graph drawing while visiting Monash University in October 2002, and encouraged us to address this problem. NICTA (National ICT Australia) is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

[1] F. Brandenburg, D. Eppstein, M. Goodrich, S. Kobourov, G. Liotta, and P. Mutzel. Selected open problems in graph drawing. In G. Liotta, editor, *Proceedings of the 11th International Symposium on Graph Drawing*, volume 2912 of *LNCS*, pages 515–539. Springer, 2003.

[2] G. Di Battista, P. Eades, R. Tamassia, and I. Tollis. *Graph Drawing: Algorithms for the visualization of graphs*. Prentice Hall, 1999.

[3] P. Eades, T. Lin, and X. Lin. Two tree drawing conventions. *International Journal of Computational Geometry and Applications*, 3:133–153, 1993.

[4] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[5] K. Supowit and E. Reingold. The complexity of drawing trees nicely. *Acta Informatica*, 18:377–392, 1982.