An Algorithm for Drawing Compound Graphs

François Bertault and Mirka Miller

Department of Computer Science and Software Engineering
University of Newcastle
Callaghan 2308 NSW Australia
{francois,mirka}@cs.newcastle.edu.au

Abstract. We present a new algorithm, called NUAGE, for drawing graphs with both adjacency and inclusion relationships between nodes, that is, compound graphs. Compound graphs are more general than classical graph models or clustered graphs. NUAGE can be applied to both directed and undirected compound graphs. It can be parameterized by classical graph drawing algorithms. NUAGE can be viewed as a method for unifying several classical algorithms within the same drawing by using the structure of the compound graph. Additionally, we present a refinement technique that can be used in conjunction with NUAGE to reduce the number of edge crossings.

1 Introduction

Most current systems for visualization of relational information are based on graphs. The objects or entities are the nodes of the graph, the relationships are edges between two nodes. The drawing of such graphs has received much attention and several algorithms dedicated to graphs with given combinatorial properties are available [1].

However, when we need to represent complicated relational information, with different kinds of relationships or with large amount of information, the classical graph model is not sufficient. Consequently, several extensions to the classical graph model have been proposed. Models using inclusion relationships between entities are particularly well suited for the representation of large amount of information because we can reduce the amount of information displayed by representing a set of nodes by the node that encompasses this set. A very general model for representing complex relationships is the higraph model [5]. Higraph model can represent inclusions, intersections and adjacency relationships but drawings algorithms for this model are difficult to design. Another commonly used model, called clustered graphs, consists of a classical graph and a recursive partition of the nodes of the graph. Clustered graphs model is well suited for graph drawing and several drawing algorithms have been presented [3,2]. An intermediate model between clustered graphs and higraphs, called compound graphs, has been introduced by Sugiyama and Misue for representing graphs with both inclusion and adjacency relationships. Algorithms for drawing compound digraphs have been proposed [8,9,6]. They are all based on an extension

J. Kratochvíl (Ed.): GD'99, LNCS 1731, pp. 197–204, 1999.

of hierarchical layered drawings of directed graphs and are ineffective for representing undirected edges. The approach taken in our algorithm, called NUAGE, is quite different and it does allow the representation of both directed and undirected compound graphs. Our algorithm can be viewed as a method for unifying several classical algorithms within the same drawing by using the structure of the compound graph.

2 Definitions

A *graph* $G = (V, E)$ is defined by a finite set V of *nodes* and a finite set E of *edges*, that is, unordered pairs (a, b) of nodes. If the pairs (a, b) are ordered then G is called a *directed graph*. If $(a, b) \in E$, a and b are said to be *adjacent* and a and b are called the *ends* of the edge. We denote $N_G^-(a) = \{b \mid (b, a) \in E\}$ and $N_G^+(a) = \{b \mid (a, b) \in E\}$. A *path* of length s between a node a_1 and a node a_s is a sequence of nodes $a_1, a_2, \ldots, a_s$ such that $(a_i, a_{i+1}) \in E$, for $i = 1, \ldots, s-1$. A *subgraph* $H = G|V'$ of a graph $G = (V, E)$, where $V' \subseteq V$, is the graph $H = (V', E')$ such that an edge $(a, b) \in E'$ if and only if $(a, b) \in E$.

A *rooted tree* $T = (V, E, r)$ is a directed graph such that for every node $a \in V$, except for the node r called the *root* of the tree, there is a unique path $Path_G(r, a)$ between r and a. For every node $a \in V$, except r, there exists a unique node $P_T(a)$, called the *parent* of a, such that $(P_T(a), a) \in E$. All the nodes on the path from the root to a node a (Except a itself) are called *ancestors* of a. For a node $a \in V$, the nodes in the set $N_T^+(a)$ are called the *children* of a. If $N_T^+(a)$ is empty, a is called a *leaf* of the tree, otherwise a is called an *internal node*. The *level* $L_T(a)$ of a node a is the length of the path between r and a, and by convention $L_T(r) = 0$.

A *compound graph* $C = (G, T)$ is defined as either a directed or an undirected graph $G = (V, E_G)$ and a rooted tree $T = (V, E_T, r)$ that share the same set of nodes, where all edges $(a, b) \in E_G$ are such that $a \notin Path_T(r, b)$ and $b \notin Path_T(r, a)$.

A *nested graph* $N = (G, T)$ is a compound graph such that:

$$\forall (a, b) \in E, P_T(a) = P_T(b). \tag{2.1}$$

Note that compound graphs, nested graphs and clustered graphs are all different models of graphs. Clustered graphs are compound graphs where edges in the graph are only between leaves of the tree. Nested graphs are compounds graphs where edges in the graph are only between children of the same parent.

3 Algorithm for Drawing Compound Graphs

In this section we present a new algorithm, called NUAGE, for the drawing of a compound graph $C = (G, T)$. We represent nodes by rectangles so that a node a is included in the rectangle that represents the node $P_T(a)$. For example, the compound graph defined by the tree and the directed graph of Fig. 1 is represented on the left of the Fig. 2. There is a 1-1 correspondence between the structure of the tree and the set of inclusions between nodes.

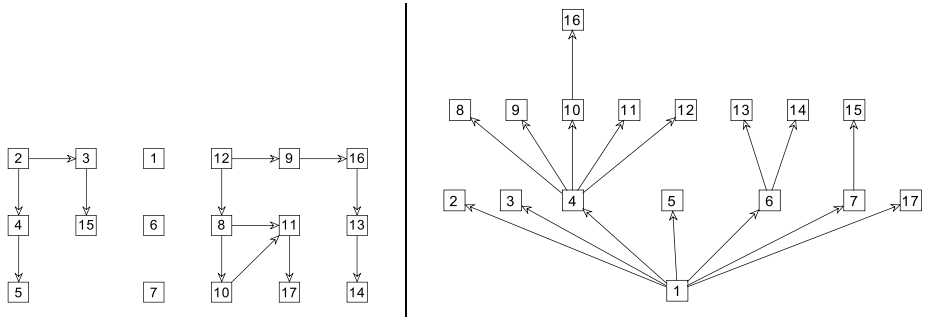

Fig. 1. Compound graph defined by a digraph and a tree.

3.1 Description of the NUAGE Algorithm

NUAGE is based on the construction of a nested graph that shares the nodes of the compound graph. The position of the nodes of the compound graph is obtained by applying an algorithm for drawing nested graphs. The construction of the nested graph is made so that the drawing of the nested graph reflects the information contained in the compound graph.

We consider a compound graph $C = (G, T)$, with $G = (V, E_G)$ and $T = (V, E_T, root)$. The steps of the NUAGE algorithm are:

Step 1. Build a nested graph $N = (H, T)$, with $H = (V, E_H)$: if $e = (a, b) \in E_g$ and $P_T(a) = P_T(b)$ then $e \in E_H$. Otherwise, e is replaced by $e' = (a', b') \in E_H$ such that a' is an ancestor of a and b' is an ancestor of b with $a' \neq b'$ and $P_T(a') = P_T(b')$.

Step 2. Apply to H an algorithm for drawing nested graphs.

3.2 Step 1: Construction of a Nested Graph Associated with a Compound Graph

The construction of the nested graph N associated with the compound graph C is based on the following principle. We start with an empty set of edges for the graph H of the nested graph. Then, for each edge (a, b) in G, we search for two distinct nodes a', an ancestor of a, and b', an ancestor of b, with $P_T(a') = P_T(b')$. We then add the edge (a', b') to H. The intuition is that if an edge influences (albeit indirectly) the position of a node v, it will also influence the position of all the nodes included in v. Thus if we replace an end v of an edge by a node which contains v, the edge will continue to have an influence on the position of v.

Algorithm (Nested graph associated with a compound graph)

- *Input:* A compound graph $C = (G, T)$, with $G = (V, E_G)$ and $T = (V, E_T, root)$.
- *Output:* A nested graph $N = (H, T)$, with $H = (V, E_H)$ and $|E_H| \leq |E_G|$.

```
N:=(H, T) with H = (V, ∅);
FOR ALL (a, b)∈G DO
   a′:=a; b′:=b;
   WHILE P_T(a′)≠P_T(b′) DO
      IF L_T(a′) = L_T(b′) THEN
         a′:=P_T(a′); b′:=P_T(b′);
      ELSE_IF L_T(a′)>L_T(b′) THEN
         a′:=P_T(a′);
      ELSE b′:=P_T(b′);
   END_WHILE;
   add edge (a′, b′) to H;
```

The graph on the right of figure 2 is an example of nested graph obtained after the application of the first step of the algorithm on the compound graph on the left of Fig. 2. The complexity of the first step of the algorithm NUAGE is $O(|E| \log |V|)$ in the average case and $O(|E||V|)$ in the worst case. Indeed, for each edge, the number of steps of the `WHILE` loop is bounded by the height of the tree T.

3.3 Step 2: Drawing of the Nested Graph

The second step of NUAGE is to draw the nested graphs associated with the compound graph. This gives us the position and the size of each node of the compound graph. The algorithm that we propose for drawing nested graphs is based on the application of classical graph drawing algorithms to each subgraph defined by the nodes with a same parent in the tree. This algorithm is called FLEUR.

We consider a nested graph $N = (G, T)$, with $G = (V, E_G)$ and $T = (V, E_T, r)$. The position of a node $v \in V$ is denoted by $(x(v), y(v))$. The rectangle that represents a node $v \in V$ is denoted $r(v)$. Given a drawing of a graph G', we can define the smallest rectangle that includes the nodes and edges of G'. We call $bb(v, G')$ the operation which assigns to $r(v)$ the smallest rectangle that includes the graph G'.

We denote by $\mathcal{A}$ the class of algorithms that can be applied to undirected or directed graphs, depending on whether we consider a directed or undirected compound graph. For each internal node of the tree of a compound graph, we define a *mode* function M which takes an internal node of the tree T as an argument and returns an algorithm in $\mathcal{A}$. For example, in Fig. 2, we chose the mode of the node 4 to be a force-directed algorithm for drawing undirected graphs [4], and the mode of nodes 1 and 6 to be an algorithm for drawing trees [7]. The mode information of a node v indicates which algorithm is to be used for drawing the subgraph formed by the nodes with v as the parent in the tree T. The function that applies the algorithm $M(v)$ is called `position_`$M(v)$. We assume that the mode algorithms have the property of avoiding overlaps between the rectangles of nodes. Alternatively, we can always apply simple post-processing, after some of the mode algorithms, to remove overlaps between the rectangles of nodes.

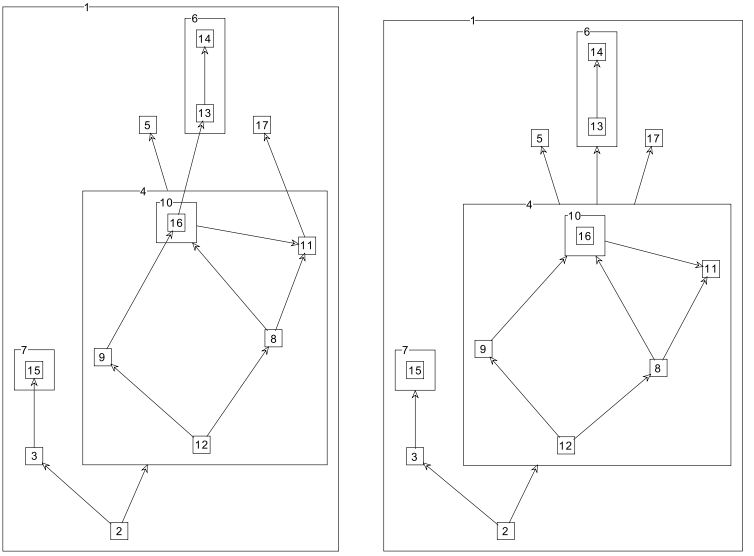

Fig. 2. Left: Drawing of the compound graph of Fig. 1. Right: Nested graph associated with the compound graph on the left.

The algorithm for drawing nested graphs is simple. We perform a depth first search traversal of the tree T and, for each internal node v of the tree, we consider the graph $H = G|_{N_T^+(v)}$, formed by the nodes with v as parent in the tree. We continue the tree traversal to calculate the width and height of the rectangles associated with the nodes in H. Then we compute the positions of the nodes in H by applying the algorithm given by $M(v)$. Next we determine the width and height of $r(v)$ by applying the $bb(v, H)$ operation. The absolute positions of the nodes in H are then set to be relative to the position of node v. After the tree traversal, we transform the relative positions of the nodes into absolute positions. The algorithm is described below.

Algorithm (Drawing of nested graphs)

- *Input:* A nested graph $N = (G, T)$, with $G = (V, E_G)$ and $T = (V, E_T, r)$. The size of the rectangle of each leaf of T is known. The mode function is defined for each internal node of T.
- *Output:* The position, width and height of each node of N is known.

```
PROCEDURE relative_position(v:node)
 IF N_T^+(v)≠∅ THEN
   FOR ALL s ∈ N_T^+(v) DO
     relative_position(s); //determine the size of rectangles of the nodes in N_T^+(v)
   H:=G|_{N_T^+(v)};
   position_M(v)(H); //we determine the positions of nodes in H
   bb(v,H); //compute the size of r(v)
   FOR ALL s ∈ N_T^+(v) DO    //the positions of nodes in N_T^+(v) are set relatively to v
     x(s):=x(s)-x(v); y(s):=y(s)-y(v);
```

```
PROCEDURE absolute_position(v:node, dx:INTEGER, dy:INTEGER)
  x(v):=x(v)+dx; y(v):=y(v)+dy;
  FOR ALL s∈N_T^+(v) DO
     absolute_position(s,x(v),y(v));

PROCEDURE FLEUR(N)
 relative_position(r); absolute_position(r,0,0);
```

The complexity of the algorithm depends on the complexity of the mode algorithms. If the overall complexity of the modes algorithms is $O(T(G))$ for a graph G, the cost of the FLEUR algorithm is given by $O(|V|+\sum_{v\in V} T(G|_{N_T^+(v)}))$. If the mode algorithms are linear in the number of nodes, the FLEUR algorithm remains linear.

At this point, we have completely specified the NUAGE algorithm which consists of the two steps described below.

Step 1. Build the nested graph H associated with the compound graph.
Step 2. Apply to H the FLEUR algorithm for drawing nested graph.

Note that if we consider one node v of the compound graph C, we can see that during the computation of the positions of the nodes $N_T^+(v)$ included in v, NUAGE simply ignores all the edges between these nodes and the nodes not included in v because these edges are ignored in the nested graph. This can lead to edge crossings that could be easily removed by exchanging positions of the nodes. By intuition, we can see that these particular nodes should be placed near the boundaries of the node v. For example, if we consider the compound graph (a) in Fig. 3, we can remove an edge crossing just by swapping the positions of nodes 5 and 17. In the next section we will consider some refinement to NUAGE.

4 Refinement Technique

We consider a refinement step which requires the following property of the mode algorithms. Given starting positions of the nodes, it is desirable for the mode algorithms to preserve the nodes' relative positions. For example, if we consider an algorithm that gives a vertical representation of a non-planar tree (i.e., a tree in which there is no fixed order between the children of a node), the order of the children of a node in the drawing should be according to their original horizontal coordinates.

Given a compound graph representation, obtained by applying the NUAGE algorithm, the refinement step modifies the initial positions of the nodes. Since the inclusion modes algorithms should keep the relative positions of nodes, the refinement step helps to reduce the edge lengths and remove edge crossings.

To implement the refinement step, we build a new graph G_r that shares the nodes of the compound graph. The initial node positions are obtained by applying a force-directed algorithm, in which we ignore the repulsion forces. The construction of the graph G_r is very similar to the construction of the associated nested graph:

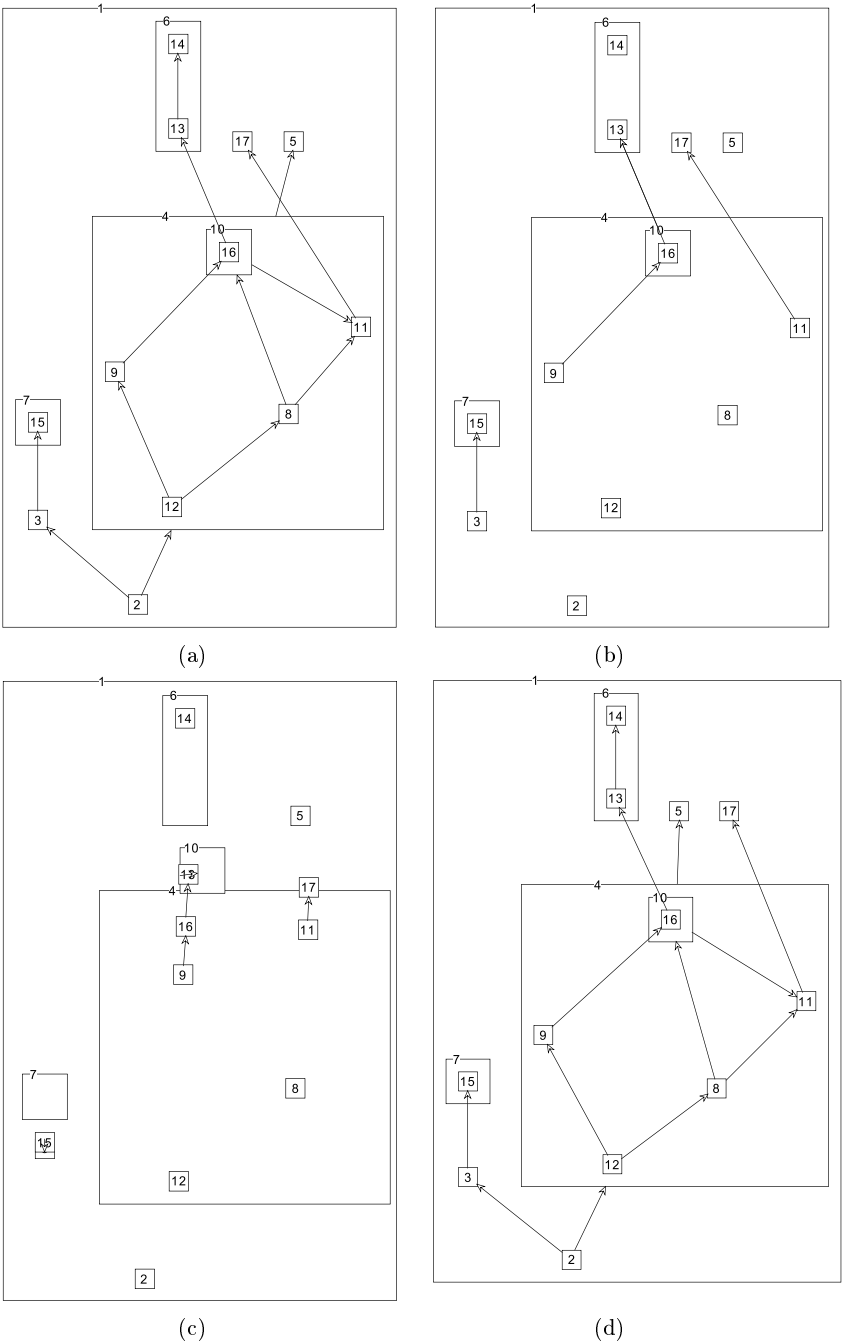

Fig. 3. Refinement step (a) the initial compound graph (b) the refinement graph associated with the compound graph (c) positions of the nodes after the contraction algorithm (d) final drawing of the compound graph.

Algorithm (Refinement graph associated with a compound graph)

- *Input:* A compound graph $C = (G, T)$, with $T = (V, E_t, r)$ and $G = (V, E_G)$.
- *Output:* The corresponding refinement graph $G_r = (V, E_r)$.

```
G_r = (V, ∅);
FOR ALL (a,b)∈E_G DO
   a':=a; b':=b;
   WHILE P_T(a')≠P_T(b') DO
      add edge (a',b') to C_r;
      IF L_T(a') = L_T(b') THEN
         a':=P_T(a'); b':=P_T(b');
      ELSE_IF L_T(a')>L_T(b') THEN
         a':=P_T(a');
      ELSE b':=P_T(b');
   END_WHILE;
```

The complexity of the refinement step is $O(|E| \log |V|)$ in the average case. Indeed, for each edge in the initial compound graph, we add $\log |V|$ edges in the refinement graph and the attraction algorithm is linear in the number of edges.

We presented an algorithm, called NUAGE, for drawing both directed and undirected compound graphs. NUAGE can be parameterized by arbitrary classical graph drawing algorithms. We implemented NUAGE in Java.

References

1. G. Di Battista, P. D. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography,. *Computational Geometry: theory and applications*, 4(5):235–282, 1994.
2. P. D. Eades and Q.-W. Feng. Multilevel visualisation of clustered graphs. In S. North, editor, *Graph drawing*, volume 1190, pages 101–111, Berkeley (USA), 1996. Springer Verlag.
3. P. D. Eades, Q.-W. Feng, and X. Lin. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. In S. North, editor, *Graph drawing*, volume 1190, pages 113–127, Berkeley (USA), 1996. Springer Verlag.
4. T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software-Practice and Experience*, 21(11):1129–1164, 1991.
5. D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988.
6. S. C. North. Drawing ranked digraphs with recursive clusters. *ALCOM Workshop on Graph Drawing 93*, 1993.
7. E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Transactions on Software Engineering*, SE-7(2):223–228, March 1981.
8. K. Sugiyama and K. Misue. Visualisation of structural information: automatic drawing of compound digraphs. *IEEE Trans. SMC*, 4(21):876–893, 1991.
9. K. Sugiyama and K. Misue. A generic compound graph visualizer/manipulator : D-Abductor. In F. J. Brandenburg, editor, *Graph Drawing*, volume 1027 of *Lecture Notes in Computer Science*, pages 500–503, Passau (Allemagne), 1995. Springer Verlag.