Clan-Based Incremental Drawing

Fwu-Shan Shieh[1] and Carolyn L. McCreary[2]

[1]Minolta-QMS, Inc. One Magnum Pass,
Mobile, AL 36618, USA
Fwu-Shan.Shieh@Minolta-QMS.com
[2]Compaq Computer Corporation, 334 South Street,
Shrewsbury, MA 01545-4112, USA
Carolyn.McCreary@compaq.com

Abstract. The stability is an essential issue for incremental drawings. To allow stable updating, means to modify graph slightly (such as adding or deleting an edge or a node) without changing the layout dramatically from previous layout. In this paper, a method for achieving stable incremental directed graph layout by using clan-based graph decomposition is described. For a given directed graph, the clan-based decomposition generates a parse tree. The parse tree, which is used for layout, is also employed in locating changes and maintaining visual stability during incremental drawing. By using the generated parse tree, each incremental update can be done very efficiently.

1 Introduction

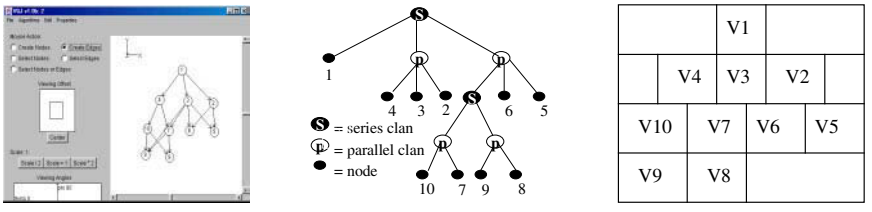
Directed graphs are an excellent means of conveying the structure and operation of many types of systems. In order to have a meaningful and understandable hand drawn graph, much time is required to plan how the graph should be organized on the page. It is especially hard to hand draw an understandable graph containing a huge number of nodes and edges. In addition, it is difficult for a user to draw a graph when the data is generated by applications (e.g., dialogue state diagrams generated by reverse engineering [1]). In the past decades, several visualization systems have been created for static (automatic) drawings [see 3 & 11 for lists]. Static drawings are not completely satisfactory because in many situations the displayed drawings are subject to change from time to time by the user (such as manual editing, browsing large graphs, and visualizing dynamic graphs) [10]. For dynamic drawings, stable incremental updating where the placement of only a minimal number of nodes and edges are modified, is essential [10, & 12]. Currently, only a few Sugiyama-based dynamic drawing systems have been developed for acyclic directed graphs [6, 10, & 15], and general directed graphs [14]. Based on the experience gained from clan-based graph drawings [8, 9, & 13], the parse tree generated by clan-based decomposition can be used to locate updates and generate stable incremental drawings easily [12].

J. Marks (Ed.): GD 2000, LNCS 1984, pp. 384-395, 2001.

2 Clan-Based Graph Drawing

Clan-based graph decomposition parses a directed acyclic graph (DAG) into a hierarchy of subgraphs. These new subgraphs generated by the decomposition are called clans and a clan is classified as one of three types: (a) **series**, (b) **parallel**, and (c) **primitive** [2, 4, and 5].

By using Clan-based graph decomposition, any digraph can be decomposed into an inclusion tree, known as the parse tree, of subgraphs (clans) whose leaves are singleton clans (graph nodes) and whose internal nodes are complex clans (series or parallel) built from their descendants. The primitive clans are decomposed into series and parallel clans by augmenting edges from all the source nodes of the primitive to the union of the children of the sources [4, 5]. After decomposition, a bounding box with computed dimension is associated with each clan and the nodes in the clan are assigned locations within the bounding box. The generated parse tree of the graph with bounding boxes attributed is used to provide geometric interpretations to the graph. To show the directed graph where the edges uniformly point downward (or upward in the case of a reverse edge), the series clans are displayed vertically and connected by inter-clan edges, and the parallel clans are displayed horizontally with no edges between them. To achieve an aesthetically pleasing layout, the nodes are centered. Figure 1 shows a graph, parse tree, and node layout. . For the details about clan-based graph drawing, please refer to [7, 8, and 9].



(a) Graph (b) Parse Tree (c) Node Layout

Fig. 1. Graph, Parse Tree, and Node Layout

3 Clan-Based Incremental Drawing

In order to have stable incremental drawing for clan-based graph decomposition, any layout computation for a successive drawing should be limited to a minimum area that contains updates only. Since parallel clans contain no inter-clan connections, this limited area for clan-based drawing is a series clan (called minimum series clan, MSC). After the MSC is identified, the layout algorithm is applied to the MSC only.

During the incremental drawing, the previous graph drawing and its corresponding parse tree, attributed with bounding boxes, are used to locate the MSC for the next graph and its drawing. The MSC contains all nodes affected by the updates except the added nodes, which are not in the previous drawing. The updates could be multiple node or edge insertions and deletions. For an update, the affected nodes include (a)

nodes added, (b) nodes deleted, (c) nodes connected to deleted nodes, (d) nodes connected to added nodes, (e) nodes connected by added edges, (f) nodes connected by deleted nodes, and (g) user selected nodes.

The following notations are used to denote graph objects in iteration i:

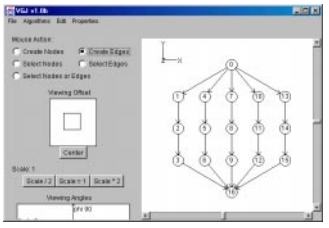
(1) G_i , graph.
(2) E_i , all edges of graph G_i.
(3) V_i, all nodes of graph G_i.
(4) D_i, drawing of graph G_i.
(5) T_i, parse tree of drawing D_i with its bounding box and position attributes.
(6) E_{add}, edges added to drawing D_{i-1}.
(7) E_{del}, edges deleted from drawing D_{i-1}.
(8) N_{add}, nodes added to drawing D_{i-1}.
(9) N_{del}, nodes deleted from drawing D_{i-1}.
(10) $N_{c\text{-}add\text{-}n}$, nodes connected to added nodes N_{add}.
(11) $N_{c\text{-}del\text{-}n}$, nodes connected to deleted nodes N_{del}.
(12) $N_{c\text{-}add\text{-}e}$, nodes connected by added edges E_{add}.
(13) $N_{c\text{-}del\text{-}e}$, nodes connected by deleted edges E_{del}.
(14) N_{sel}, selected nodes.
(15) $N_{affected}$, nodes affected by update. $N_{affected} = N_{add} \cup N_{del} \cup N_{c\text{-}add\text{-}n} \cup N_{c\text{-}del\text{-}n} \cup N_{c\text{-}add\text{-}e} \cup N_{c\text{-}del\text{-}n} \cup N_{sel}$.
(16) P_{i-1}, array of clan tree pointers to leaf nodes of T_{i-1}.

The **msc** and **act** subscripts denote graph objects of the minimum series clan and the subgraph that requires layout computation, respectively.

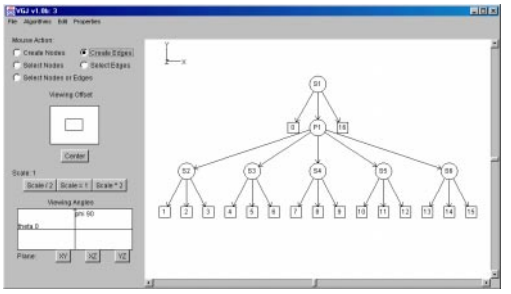
When few changes are made from one iteration of the graph to the next, the new graph can be drawn by:

(1) identifying the MSC, C_{msc}, and its corresponding graph, G_{msc},
(2) adding/deleting nodes and edges from $G_{msc,}$ to form the affected graph, G_{act},
(3) computing the parse tree of G_{act},
(4) determining the layout of G_{act} from the parse tree, and
(5) scaling G_{act} to fit in the space occupied by G_{msc}.

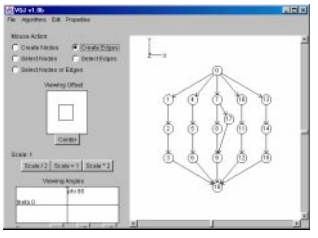
Figure 2 shows a graph drawing, parse tree, and its updated graph. The affected nodes $N_{affected}$ for this update are nodes 7, 9, and 17. From Figure 2 (b) parse tree, the MSC that contains $N_{affected}$ - N_{add} is series clan S4. For Figure 2 (a), the G_{msc} consists of nodes (7, 8, 9) and edges ((7, 8), (8, 9)). After G_{msc} is found, the clan-based layout algorithm will be applied only to sub-graph G_{act} of current graph. The sub-graph G_{act} can be identified as G_{act}.nodes = G_{msc}.nodes $\cup$ N_{add} - N_{del} and G_{act}.edges = G_{msc}.edges $\cup$ E_{add} - $E_{del.}$ In Figure 2 (c), the G_{act} consists of nodes (7, 8, 9, 17) and edges ((7, 8), (8, 9), (7, 17), (19, 9)). After the layout algorithm is applied to G_{act}, the parse tree T_{act} and drawing D_{act} are generated (as shown on Figure 3). The size and position of G_{act}'s drawing D_{act} are attributed in parse tree T_{act}.



(a) Graph Drawing

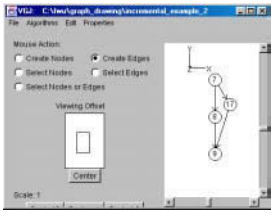


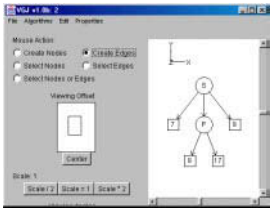
(b) Parse Tree for (a)

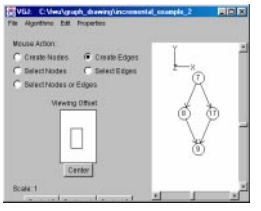


(c) Node 17 and Edges (7, 17) & (17, 9) Added to (a)

Fig. 2. Graph Drawing, Parse Tree, and Updated Graph





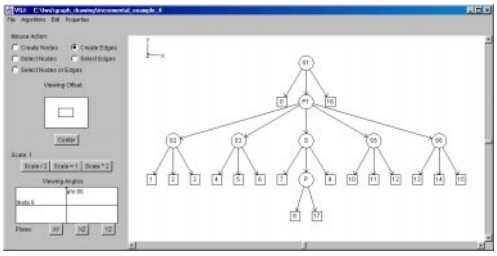


(a) Graph G_{act} of Fig. 2 (c) (b) Parse Tree of graph D_{act} (c) Drawing of Graph G_{act}

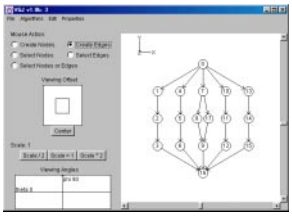
Fig. 3. Graph G_{act}, Parse Tree, and Drawing

In order to minimize the number of nodes that must be moved for stable incremental drawing, only the G_{act} is recomputed for current graph G_i, and the drawing D_{act} of G_{act} is sized to be contained in the area used by G_{msc}'s drawing D_{msc}. The size and position of G_{msc}'s drawing D_{msc} are attributed in C_{msc}. If the size of D_{act} is greater than D_{msc}, the D_{act} is scaled down. If the size of D_{act} is smaller than D_{msc}, the D_{act} is positioned in the center of the area used by D_{msc}.

The MSC C_{msc} in T_{i-1} is replaced by graph G_{act}'s parse tree T_{act}. The modified parse tree becomes the current parse tree T_i for current graph G_i. Figure 4 shows the new parse tree T_i and it's drawing D_i of Figure 2 (c)'s incremental update.

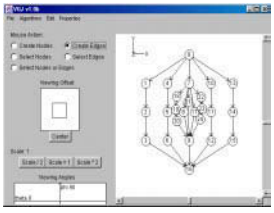


(a) Parse Tree from Fig. 2(b) with Series Clan S4 Replaced

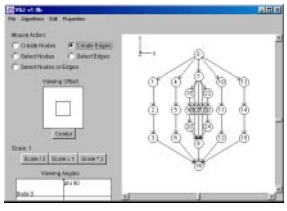


(b) Drawing of Figure 2(c)

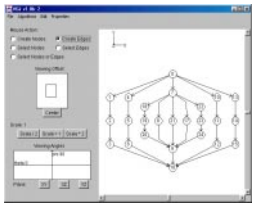
Fig. 4. Drawing for Figure 2 (c)'s Incremental Update



(a) Updated Graph of Figure 4(b)



(b) Drawing Shows Nodes Overlapped



(c) New Drawing of (b)

Fig. 5. An Updated Graph and Drawings for Fig. 4 (b)

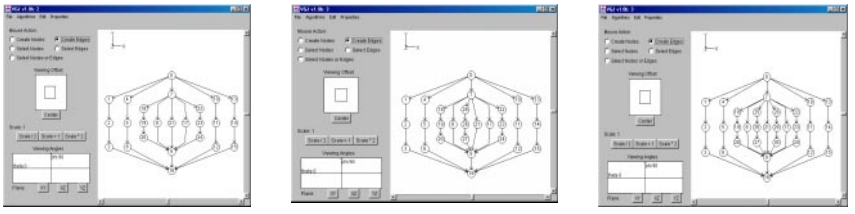
4 Insuring Readability

When D_{act} is scaled down to fit in the area used by D_{msc}, the scaled drawing might be unreadable. Figure 5 (b) shows this problem after an update. Figure 5 (a) is an updated graph of Figure 4 (b)'s drawing. In this updated graph, nodes (18, 19, 20, 21, 22, 23, 24) and edges ((7, 18), (18, 19), (19, 20), (20, 9), (7, 21), (21, 9), (7, 22), (22, 23), (23, 24), (24, 9)) are added. Figure 5 (b) is the drawing for Figure 5 (a). In the Figure 5 (b), the scaled down drawing D_{act} has overlapping nodes that make it a problem for the user to read the drawing. So, a maximum scale limit is needed to ensure readability. Figure 5 (c) shows new drawing of Figure 5 (a). In Figure 5 (c), the scale limit is applied. After the scale limit is reached, the D_{act} is given more space to maintain the readability. Although only the subtree rooted at MSC is replaced, the bounding box and placement attributes of the entire parse tree must be computed. For the example in Figure 5, this has the effect of moving nodes (1, 2, 3, 4, 5, and 6) to the left and nodes (10, 11, 12, 13, 14, and 15) to the right.

5 Locality of Incremental Drawing

Successive modifications to graphs often occur within a small geometric or logical neighborhood [10]. In order to build more stable drawing with fewer computations, the algorithms should take advantage of this locality property. When adjusting the previous clan tree T_{i-1} and the previous drawing D_{i-1} to make extra space for D_{act}, if the space created is the exact space needed by D_{act}, then any nodes added to the current G_{act} might cause T_i and D_i to be adjusted again. If the graph is expected to grow, the scale factor for G_{act} should be increased.

Figure 6 shows an example of extra space created for future updates. Figures 6 (b) is successive drawings of Figure 6 (a) and Figure 6 (c) is successive drawing of Figure (b). In Figures 6 (b) and 6 (c), nodes (1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15) are not re-positioned because the possible extra space needed by successive drawings has been generated during Figure 6 (a) drawing layout computation.



(a) New Drawing of Fig. 5 (b) (b) Successive Drawing of (a) (c) Successive Drawing of (b)

Fig. 6. New Update Drawing of Figure 5 (c) with Nodes and Edges Added

6 Reposition Nodes Not in the Minimum Series Clan

If more space is needed, what nodes need be re-positioned or re-sized? Let $P_{msc\text{-}root}$ be the path from C_{msc} to root in previous parse tree T_{i-1}, and L_{adjust} be extra length and W_{adjust} be extra width needed by D_{act}. To make extra space, for each clan C_{path} along the path $P_{msc\text{-}root}$ needs to add W_{adjust} to C_{path}'s width and L_{adjust} to C_{path}'s length. Also, along the path $P_{msc\text{-}root}$,

1. all left siblings of C_{path} whose parent is a parallel clan need to be shifted left with W_{adjust} / 2 distance,
2. all right siblings of C_{path} whose parent is a parallel clan need to be shifted right with W_{adjust} / 2 distance, and
3. all right siblings of C_{path} whose parent is a series clan need to be shifted down with L_{adjust} distance.

Figure 7 is a parse tree with C_{msc} identified. The path $P_{msc\text{-}root}$ includes clans C_{msc}, P7, S5, P2, and S1. In this parse tree, if more space is needed :

1. node 64 and clan S4's nodes need to be shifted to the left,
2. clan S10's nodes and clan S6's nodes need to be shifted to the right, and
3. and node 65, clan P8's nodes, and P3's nodes need to be shifted down.

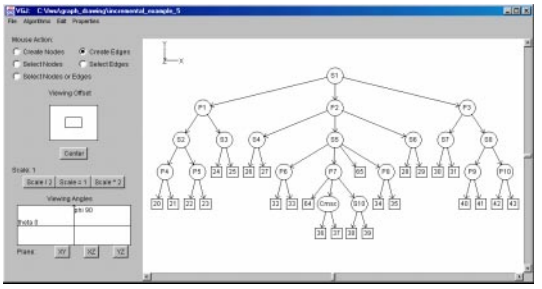
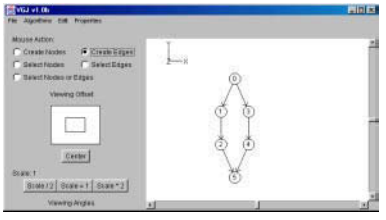


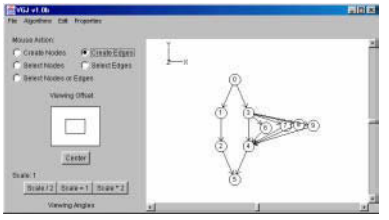
Fig. 7. Parse Tree with C_{msc} identified

7 Resizing Affected Clans

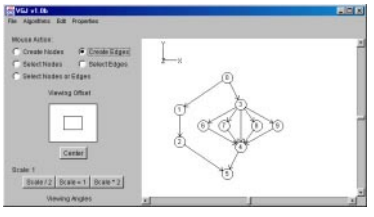
Is it necessary to resize all clans along the path $P_{msc\text{-}root}$? For a parallel clan, its length is the max value of all children's lengths. For a series clan, its width is the max value of all children's widths. For each clan C_{path} along the path $P_{msc\text{-}root}$, if both the C_{path} 's width and length do not exceed parent's max values, it is not necessary to resize parent.



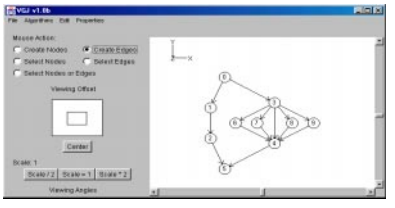
(a) Drawing



(b) Updated Graph of Drawing (a)



(c) Drawing for (b)

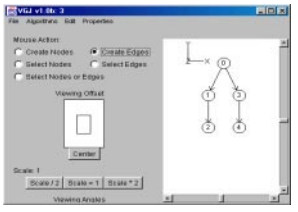


(d) New Drawing of (b)

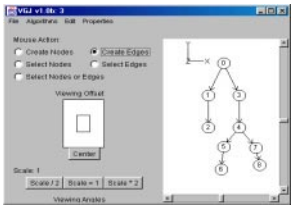
Fig. 8. Updated Graph and Drawings

In some situations, it is not necessary to re-position siblings along the path $P_{msc\text{-}root}$ or resize clans which contains C_{msc} when the drawing D_i requires extra space. In Figure 8 (b), nodes (6, 7, 8, 9) and edges ((3, 6), (6, 4), (3, 7), (7, 4), (3, 8), (8, 4), (3, 9), (9, 4)) are added to Figure 8 (a) drawing. Figure 8 (c) is the drawing for Figure 8 (b). In Figure 8 (c), nodes 1 and 2 are shifted to the left to make some space for updates. However, the shift is not necessary. Since there are no nodes at the right hand side of nodes 3 and 4, it would be more reasonable for updates to grow toward

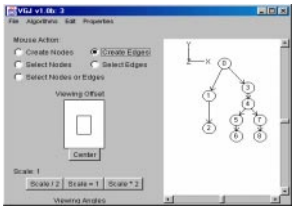
the right without shifting nodes 1 and 2. Figure 8 (d) shows a different drawing of Figure 8 (b). In Figure 8 (d), nodes 1 and 2 are not moved.



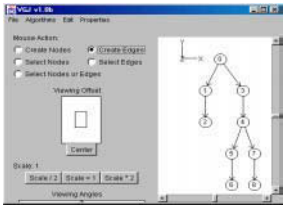
(a) Drawing



(b) Updated Graph of Drawing (a)



(c) Drawing for (b)



(d) New Drawing of (b)

Fig. 9. Updated Graph and Drawings

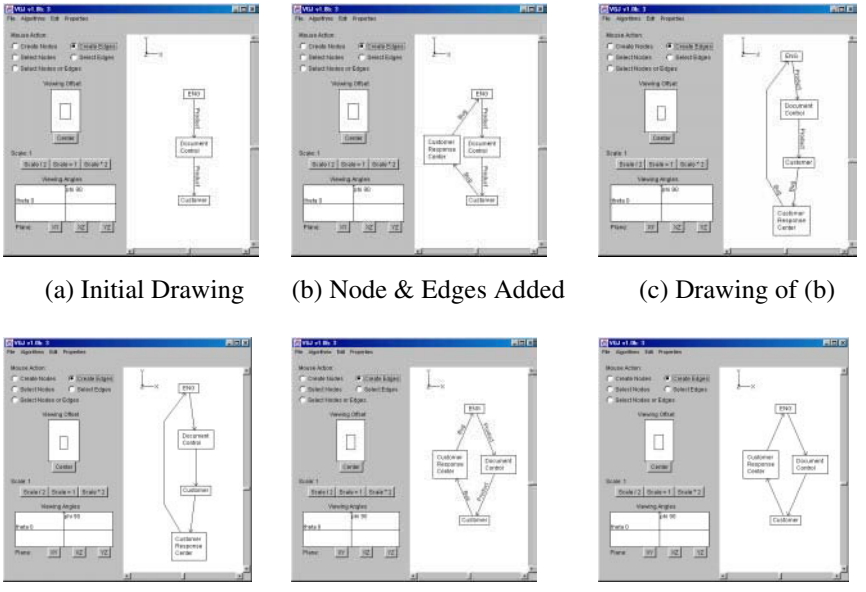
Figure 9 shows a case when re-sizing is not necessary. In Figure 9 (b), nodes are appended to the node 4 of Figure 9 (a). Figure 9 (c) is the drawing for Figure 9 (b). In Figure 9 (c), the sub-graph containing nodes (3, 4, 5, 6, 7, and 8) is scaled even when there are no nodes below them. In this case, the graph should be able to grow downward freely without affecting the drawing stability. Figure 9 (d) shows a different drawing of Figure 9 (b). The path $P_{msc\text{-}root}$ can be used to identify the cases where re-sizing / re-positioning is not necessary:

1. along the path $P_{msc\text{-}root}$, if there are no left siblings for a clan whose parent is a parallel clan, the drawing D_i can grow toward left without shifting other clans to the right,
2. along the path $P_{msc\text{-}root}$, if there are no right siblings for a clan whose parent is a parallel clan, the drawing D_i can grow toward right without shifting other clans to the left, and
3. along the path $P_{msc\text{-}root}$, if there are no right siblings for a clan whose parent is a series clan, the drawing D_i can grow downward without scaling down length.

8 Incremental Drawing of Cyclic Graphs

Using the depth first search to find an edge to be reversed for the cyclic graph may not be suitable for some applications [12]. Figure 10 shows a problem in incremental drawing when only one edge is drawn upward for each cycle of cyclic graphs. Figure 10 (a) shows a path of a company's system code release. In Figure 10 (b), a new path for bugs report is added. Figure 10 (c) is the drawing of Figure 10 (b). The Figure 10 (c) does not represent the two paths in the expected way. If edge labels are removed (as Figure 10 (d)), it will be even more difficult for the user to visualize the concept of

two different paths. In order to improve this situation, the visual input graph nodes' position information is used. During the graph layout computation, if upward edges contributed to a cycle are found, those edges will be reversed for layout computation and then be changed back to upward after layout computations done. Figure 10 (e) shows new drawings of Figure 10 (b). As shown in the Figure 10 (f), even when the edges are not labeled, the drawing still represents the concept of two paths.

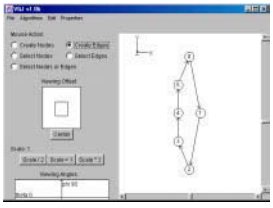


(a) Initial Drawing (b) Node & Edges Added (c) Drawing of (b)

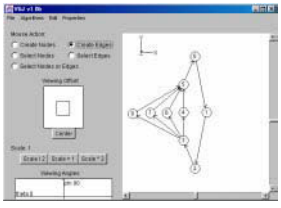
(d) Drawing of (c) with Labels Removed (e) Drawing of (b) (f) Drawing of (b) with Label Removed

Fig. 10. Updated Graph and Drawings of Incremental Cyclic Drawing

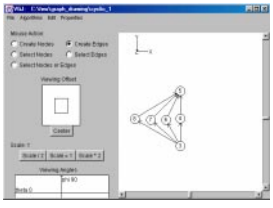
By using clan-based drawing for incremental drawing, the layout computation only applies to the affected sub-graph G_{act} instead of the entire graph. Sometimes when the sub-graph is extracted from the entire graph, its original role in the entire graph might be missing. The Figure 11 (b) is an update of Figure 11 (a), and Figure 11 (c) is Figure 11 (b)'s G_{act} which requires layout computation. The G_{act} is not a cyclic graph, but G_{act} is part of the cycles of Figure 11 (b) cyclic graph originally. Since G_{act} is not a cyclic graph, no edges are reversed during the layout computation. Figure 11 (d) shows the incorrect updated drawing of Figure 11 (b). To insure proper edge direction, upward edges need be checked to see if those edges are part of cycles of original graph. If an upward edge contributes to a cycle, this upward edge will be reversed during the computation and reversed again after the computation. Figure 11 (e) shows the correct drawing of Figure 11 (b).



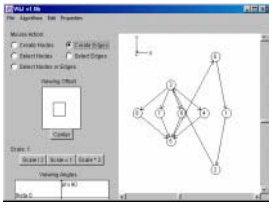
(a) Initial Drawing



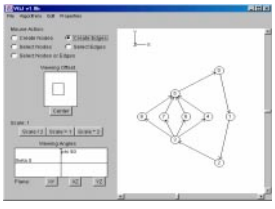
(b) Nodes & Edges Added



(c) Sub-graph of (b)



(d) Incorrect Drawing of (b)



(e) Correct Drawing of (b)

Fig. 11. Drawing and Graph with Upward Edges Added

9 Incremental Drawing Examples

Figure 12 shows a series of incremental drawings created by the clan-based drawing algorithm. The Figures 12 (b), (e), and (f), show that some upward paths were added. Those upward paths are part of cycles. After nodes (11, 12, 13) and edges ((4, 11), (11, 12), (12, 13), (13, 6)) are added to Figure 12 (f), the length of new drawing, Figure 12 (g), is not changed because the minimum affected area still has enough space for updates. Figures 13 (n) is the new drawing of 12 (m) after upward edges ((7, 18), (7, 20)) are added. The edges ((7, 18), (7, 20)) are not part of cycles, so node 7 is moved to above nodes 18 and 20.

10 Conclusion

Clan-based graph drawing using a parse tree improves the incremental drawing stability for directed graph drawings in several areas:

(1) The attributed parse tree provides an easy way to layout graphs with nodes of different sizes. A node can be spanned over more than one level in drawing if necessary. During the incremental updates, if the change is only to enlarge nodes, the new updates might be done very easily without re-positioning other nodes.

(2) The utilization of parse trees allows the incremental drawing to be stable without sacrificing aesthetic criteria and speed. By using clan-based drawing, the incremental drawing can be done very efficiently because the changed and recomputed area can be localized with the provided parse tree.

(3) No extra constraints are needed to maintain incremental drawing stability. For clan-based drawing, the node position constraints are embedded in the parse tree.
(4) By using the parse tree, it is easy to determine whether the modifications are in the interior or the exterior boundary area of a drawing. If the changes are made to exterior area, the updates can grow freely toward the open area without re-positioning other nodes.
(5) By using the parse tree to locate the minimum affected area of updates, the locality issue can be considered for the future stable updates.

References

1. J. H. Cross II and R. S. Dannelly, "Reverse Engineering Graphical Representations of X Source Code," International Journal of Software Engineering and Knowledge Engineering, Spring, 1996.
2. A. H. Deutz, A. Ehrenfeucht, G. Rozenberg, "Clans and regions in 2-structures," Theoretical Computer Science, 129, 207-262, 1994.
3. G. Di Battista, P. Eades, R. Tamassia, I. Tollis, "Algorithms for Drawing Graphs: an Annotated Bibliography", Computation Geometry: Theory and Applications, 4(5):235-282, 1994.
4. A. Ehrenfeucht and G. Rozenberg, "Theory of 2-Structures, Part I: Clans, Basic Subclasses, and Morphisms," Theoretical Computer Science, Vol. 70, 277-303, 1990.
5. A. Ehrenfeucht and G. Rozenberg, "Theory of 2-Structures, Part II: Representation Through Labeled Tree Families," Theoretical Computer Science, Vol. 70, 305-342, 1990.
6. M. Frohlich,"Incremental Graphout in Visualization System – daVinci," PhD thesis, Department of Computer Science, The University of Bremen, Germany, November 1997.
7. C. M. McCreary, R. O. Chapman, and F. S. Shieh, "Using Graph Paring for Automatic Graph Drawing", IEEE Trans. on Systems Man, and Cybernetics -- Part A: Systems and Humans, Vol. 28, No. 5, 545-561, 1998.
8. C. L. McCreary and A. Reed, "A Graph Parsing Algorithm and Implementation," Tech. Rpt. TR-93-04, Dept. of Comp. Sci and Eng., Auburn U. 1993.
9. C. McCreary, F. S. Shieh, and H. Gill, "CG: a Graph Drawing System Using Graph-Grammar Parsing," Lecture Notes in Computer Science, Vol. 894, 270-273, Springer-Verlag, 1995.
10. S. C. North, "Incremental Layout in DynaDAG," Lecture Notes in Computer Science, Vol. 1027, 409 - 418, Springer-Verlag, 1996.
11. G. Sander, "Graph Drawing Tools and Related Work," http://www.cs.uni-sb.de/RW/users/sander/html/gstools.html
12. F. S. Shieh, "Stability and Topology of Graph Drawing," Auburn University, Ph.D. dissertation, 2000.
13. F. S. Shieh, and C. L. McCreary, "Directed Graphs Drawing by Clan-based Decomposition," Lecture Notes in Computer Science, Vol. 1027, 472 - 482, Springer-Verlag, 1996.
14. Tom Sawyer, "*Graph Toolkit"*. http://www.tomsawyer.com.
15. K. Sugiyama, S. Tagawa and M. Toda, "Methods for Understanding of Hierarchical System Structures," IEEE Trans. on Sys. Man, and Cyb., SMC-11, 109-125, 1981.

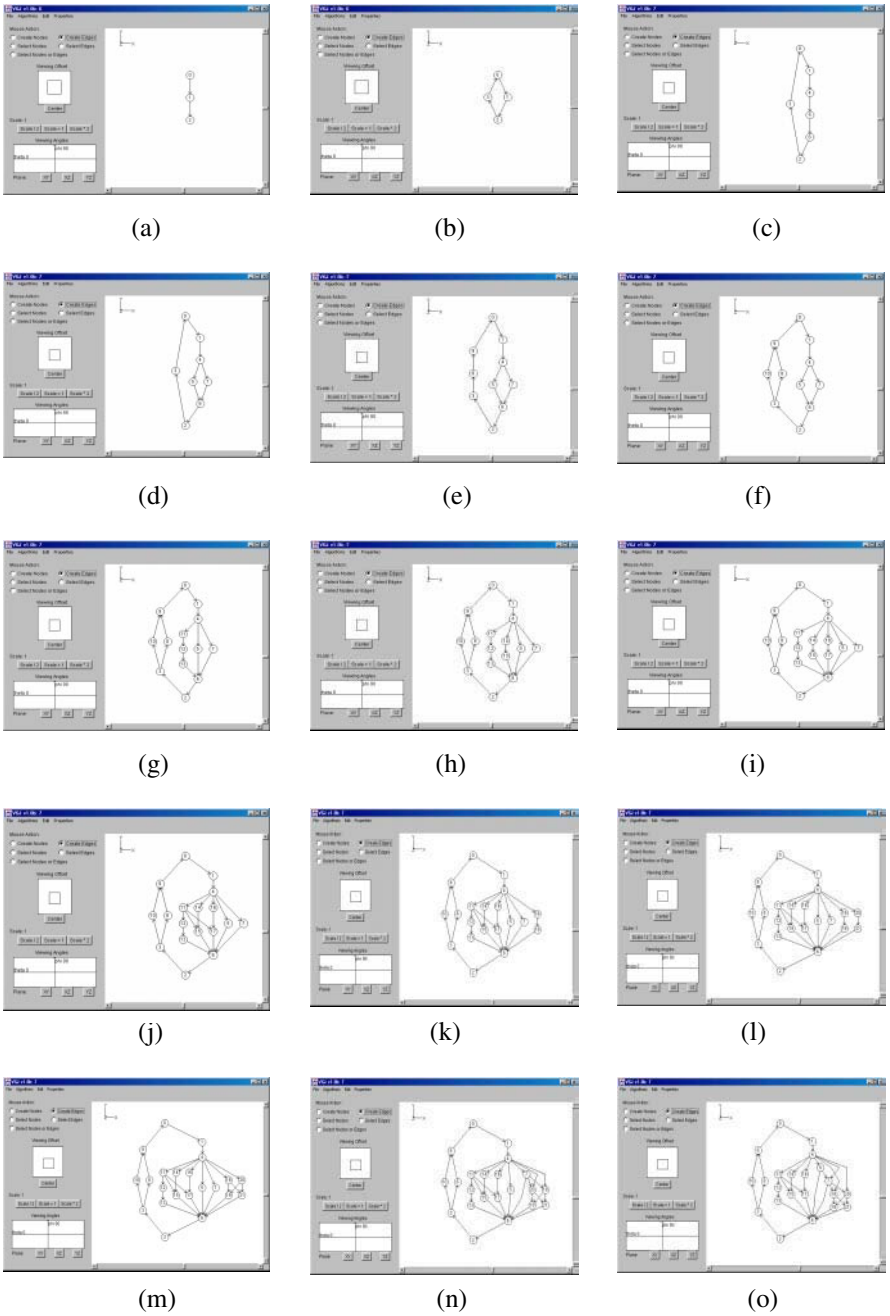


Fig. 12. Clan-Based Incremental Drawings