Visualization of Clustered Directed Acyclic Graphs with Node Interleaving

Pushpa Kumar
University of Texas at Dallas
Richardson, TX 75080
pkumar@utdallas.edu

Kang Zhang
University of Texas at Dallas
Richardson, TX 75080
kzhang@utdallas.edu

ABSTRACT

Graph drawing and visualization represent structural information as diagrams of abstract graphs and networks. An important subset of graphs is directed acyclic graphs (DAGs). *E-Spring* algorithm, extended from the popular spring embedder model, eliminates node overlaps in clustered DAGs by modeling nodes as charged particles whose repulsion is controlled by edges modeled as springs. The drawing process needs to reach a stable state when the average distances of separation between nodes are near optimal. This paper presents an enhancement to E-Spring to introduce a stopping condition, which reduces equilibrium distances between nodes and therefore results in a significantly reduced area for DAG visualization. It imposes an upper bound on the repulsive forces between nodes based on graph geometry. The algorithm employs node interleaving to eliminate any residual node overlaps. These new techniques have been validated by visualizing eBay buyer-seller relationships and resulted in overall area reductions in the range of 45% to 79%.

Categories and Subject Descriptors
E.1 [Data]: Data Structures – Graphs and networks.
H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces – Graphical user interfaces (GUI).

General Terms
Algorithms, visualization, graph drawing

Keywords
Node interleaving, directed acyclic graphs, overlapping nodes

1. INTRODUCTION

Graph drawing and visualization is an area of extensive research in recent years. Applications of graph visualization include genealogy, cartography, sociology, software engineering, VLSI design, and visual analysis. Cluttered drawings of graphs generally produce undesirable visualizations. Various criteria for determining the graph quality have been proposed [1][2][3]. A graph drawing is sometimes measured by how well it meets the following aesthetic criteria [4][13][[14]:

- Minimization of *edge crossings*
- Minimization of *area*: Visual space is often at a premium and reducing the overall area of the bounding box of the graph is often desirable. The shape of the graph can also be important, so that it has a certain aspect ratio, hence fitting well on a page or screen.
- Maximization of *symmetry*
- Minimization of *total edge length*, uniform *edge length*: minimization of the sum of the length of the edges should cause a reduction of the area of the graph.
- *Aspect ratio* close to a specified value
- *Node Separation*: Nodes should be sufficiently far apart from their nearest neighbor to be easily distinguished and to avoid occlusion.
- *Readability*: Labels must have legible sizes.
- Avoidance of *Overlap*: Labels should not overlap with other labels or other graphical features.

Various algorithms [2][3] have been proposed for removing node overlaps in graphs. Being most widely used, Spring algorithms [12] are a class of algorithms for drawing graphs in an aesthetically pleasing way by positioning the nodes of a graph in two or three dimensional space so that all the edges are of more or less equal length, and there are as few crossing edges as possible. They assign forces as if the edges were springs following Hooke's law, the forces are applied to the nodes, pulling them closer together or pushing them further apart. This is repeated iteratively until the drawing process comes to an equilibrium state. This process is typically animated so that the viewer's mental map is preserved [25]. At equilibrium, node overlap is gradually reduced [4][5][6]. Spring algorithms are regarded as effective tools for visualizing undirected graphs. One major feature of applying spring algorithms is to display symmetric properties of graphs [7]. The combined method of constructing a globally nice layout using the Kamada-Kawai [8] method and local beautification using the modified (or generalized) spring method has been proposed for non-uniform nodes.

Directed acyclic graphs (DAGs) have been used to visualize various eBusiness and other applications [10][11]. E-Spring algorithm [9] uses animation for drawing clustered DAGs without node overlap to preserve the user's mental map [25]. The animation needs to reach a stable state when the average distances of separation between nodes are near optimal. This paper introduces a stopping condition to E-Spring that significantly reduces equilibrium distances between nodes for the clustered DAG visualization. The main contribution of this paper is a novel method to eliminate node overlaps that achieves minimization of *bounded area* of DAGs. Further optimization is achieved using *node interleaving* to remove residual node overlaps. The resultant graphs obtained satisfy various aesthetic criteria like overall symmetry and readability. Experimental analysis and visualization of this method for eBay buyer-seller relationships [15] are reported, and show overall DAG area reductions in the range of 45% to 79%. The rest of this paper is organized as follows. Section 2 describes related work. Section 3 presents the E-Spring algorithm, stop condition, and further optimization. Section 4 provides details of our implementation and experimental results. Finally, a conclusion is given in Section 5.

2. RELATED WORK

We review related work on removing node overlap, determining stopping condition, reduction of area, and reducing graph visual complexity. The family of spring algorithms includes force scan (FS) [25], force transfer (FT) [6], dynamic natural length spring (DNLS) and orthogonal dynamic natural length spring (ODNLS) [4] which modify attractive and/or repulsive spring forces between nodes to achieve node overlap removal. DNLS is designed for visualization of static graph drawings while ODNLS is for dynamic graph drawing. This method produces favorable results over the force scan (FS) method.

A fast algorithm (FADE) for two dimensional drawing, geometric clustering and multilevel viewing of large undirected graphs is presented in [5]. The decomposition tree provides a systematic way to determine the degree of closeness between nodes without explicitly calculating the distance between each node. Various heuristics to minimize the area overhead for delay-driven clustering of combinational circuits under I/O, and area constraints are presented in [17]. Clustering which aims at minimal path delay is performed, and several set covering heuristics are proposed to minimize the number of clusters while satisfying the constraints and preserve the timing. A hierarchical clustering method has also been proposed in this paper to explore global area-delay tradeoff.

Caliph & Emir [19] is a pair of applications that is used for the annotation and search of digital photos focusing on semantic descriptions, and a new stop condition lets the algorithm itself decide how many iterations have to be taken to gain a force equilibrium. The attracting node in the centre results in a more circular layout for non-connected graphs. In VLSI wire routing, it is often desirable to maximize the distance between different wires. Maximizing the distance between wires is equivalent to finding the drawing in which the edges are drawn as thick as possible, i.e., allowing the graph to grow as fat as possible. The two types of stopping conditions for fat graphs are collision of two vertices and collision of two elbows [20].

An approach for practical graph layout, layout adjustment for removing overlapping in images and node-edge intersections, and boundary detection for ensuring that a diagram fits in a viewing area is presented in [16]. The solution proposed is to adjust a diagram by removing overlapping nodes and node-edge intersections; if the adjusted diagram exceeds the viewing area, some sub-diagrams become invisible. Density functions, derived visualization, navigation, and clustering techniques are used in exploring large graphs in [18]. In the automatic case, the stop condition for the subdivision can be expressed in terms of the density function, and the stop condition halts the process when the density dependent size of the cluster becomes small.

The clustered graph model is used to reduce data complexity, while a navigation model is proposed to solve the visual complexity. To assist the navigation model, the semantic fisheye view of a clustered graph is proposed as a logic abstraction strategy, and a layout algorithm is introduced for generating visualization of clustered graph in dynamic environments [21]. Generalized fisheye views are described in [22]. These views provide a balance of local detail and global context by trading off a priori importance against distance.

Some two dimensional plane drawing algorithms for clustered graphs; extending two dimensional plane drawings to three dimensional multilevel drawings by considering convex drawings and orthogonal rectangular drawings are presented in [23]. Survey on graph visualization and navigation techniques, as used in information visualization is presented in [24]. The ability to visualize and navigate potentially large, abstract graphs is often a crucial part of an application, and this survey approaches the results of traditional graph drawing from a different perspective.

3. E-SPRING ALGORITHM

In this section, we briefly review the forces involved in E-Spring algorithm, then describe techniques for removing node overlaps.

3.1 Idea of Forces

E-Spring algorithm [9] was proposed to eliminate node overlaps in clustered DAGs. The attractive forces on the springs, and the repulsive forces between the positive charges act together to generate a drawing free of node overlaps, when the system reaches a state of equilibrium. However, the nodes at equilibrium are apart at unnecessarily large distances, leading to inefficient utilization of screen space for the overall graph. The following subsection introduces a stop condition that stops the animated drawing when the distances between nodes are enough to eliminate node overlaps. This results in a significantly reduced area for the clustered DAG visualization at equilibrium.

3.2 Introducing a Stop Condition

Consider a directed acyclic graph $G_D(V, E)$ with a set of nodes $V = \{1,2,\ldots,|V|\}$, and the set of edges $E \subseteq V \times V$. We aim to minimize separation between nodes in any cluster $C = \{1,2,\ldots,|C|)$, as well as DAG inter-cluster distance to achieve bounded expansion of directed acyclic graphs. A *stop condition* is introduced that utilizes graph geometry to determine upper bound on graph expansion during animated drawing. Graph expansion automatically stops when the condition is met. The condition is derived from minimizing the sum of edge lengths within a DAG cluster, and the DAG inter-cluster distance, while providing sufficient nearest neighbor distance of separation, to prevent overlap. Following definitions and theorems establish mathematical basis for the stop condition.

Definition 1: Consider a directed acyclic graph $G_D(V, E)$ with a set of nodes $V = \{1,2,\ldots,|V|\}$, and the set of edges $E \subseteq V \times V$. Each source node is a parent node and belongs to the set of parent nodes $P_N \subset V$, a node having no children is a leaf node belonging to the set $n_N \in \{C_N\}$ where $C_N \subset V$ is the set of all child nodes. A parent node together with all its direct child leaf nodes forms a *cluster*.

Theorem 1: In a DAG drawn using the *E-Spring* algorithm, all child leaf nodes belonging to a cluster are equally distant from the parent node.

Proof: When E-Spring algorithm is applied, the equilibrium distance between a parent P_i *(i = 1…k)* and child leaf nodes n_j *(j = 1…m)* is reached when attractive forces 'F_s' on the springs balance the repulsive forces 'F_c' between the positive charges. Since all children have same charge '*q*', the edge length '*l*' between P_i and each n_j within a DAG cluster is equal.

Theorem 2: In a DAG drawn using the *E-Spring* algorithm, all child leaf nodes belonging to a cluster subtend equal angles with the parent node.

Proof: Since all child leaf nodes n_j *(j = 1…m)* have the same charge '*q*' in a cluster, they repel each other equally, and therefore are separated from each other by equal distance. By *Theorem 1* since they are equally far apart from their parent P_i *(i = 1…k)*, all child leaf nodes n_j *(j = 1…m)* in a cluster will lie along the circumference of the circle with circle radius 'C_r' equal to edge length '*l*'. Therefore all child leaf nodes will subtend equal angles 'θ' with the parent node.

Definition 2: The minimum distance of separation '*d*' between two neighboring child leaf nodes n_i and n_j, is the minimum horizontal distance between nodes *i* and *j* required to prevent them from overlapping.

Theorem 3: In a DAG drawn using the *E-Spring* algorithm, the minimum distance of separation between neighbor child leaf nodes is $2l\sin(180^\circ/N)$.

Proof: Consider 'N' number of child leaf nodes in a cluster which are not points (n_1, n_2, n_N), and have uniform string label length as shown in Figure 1a.

By *Theorem 2*, separation angle $\theta = 360^\circ/N$

$$2\alpha + \theta = 180^\circ \Rightarrow 2\alpha = 180^\circ - \theta \Rightarrow \alpha = 90^\circ - 180^\circ/N$$

$$\cos\alpha = \frac{a}{l} \qquad \therefore a = l\cos\alpha$$

∴ Minimum neighbor distance between two child leaf nodes

$= 2a = 2l\cos\alpha$

$$= 2l\cos\left(90^\circ - \frac{180^\circ}{N}\right) = 2l\sin\left(\frac{180^\circ}{N}\right) \qquad (1)$$

Theorem 4: In a DAG drawn using the *E-Spring* algorithm, the average edge length within a cluster with uniformly labeled nodes to achieve minimum separation distance between adjacent child leaf nodes is given by $(n_w + \delta / 2\sin(180^\circ/N))$.

Proof: To remove node overlap, the minimum neighboring distance between two child leaf nodes as determined by *Theorem 3*, $2l\sin(180^\circ/N) >$ node width $n_w + \delta$, where δ = tolerance.

∴ Average edge length (l_e) within a cluster to achieve minimum separation distance is given by

$$(n_w + \delta / 2\sin(180^\circ/N)) \qquad (2)$$

We now determine the condition for minimizing the distance of DAG inter-cluster node separation. Consider two DAG clusters of a directed acyclic graph as shown in Figure 1b. The parent node location for Cluster$_1$ is given by co-ordinates (x_1, y_1), while the parent node location for Cluster$_2$ is given by (x_2, y_2). The minimum inter-cluster separation between the parent nodes in two neighboring DAG clusters is then given by:

$$D_{inter\text{-}cluster} > l_{cl1} + l_{cl2} + \delta \qquad (3)$$

where l_{cl1} = average edge length of children leaf nodes of Cluster 1, l_{cl2} = average edge length of children leaf nodes of Cluster 2, $D_{inter\text{-}cluster} = \sqrt{(|x_1 - x_2|)^2 + (|y_1 - y_2|)^2}$, and δ = tolerance.

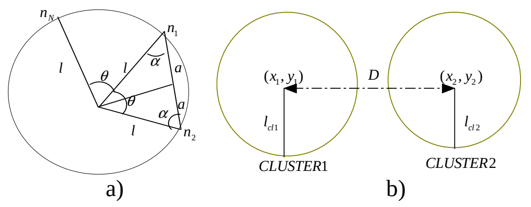


Figure 1. Edge length (a) and DAG inter-cluster distance (b).

Formulas 2 and 3 determine the stop condition for the DAG expansion, which achieves node overlap removal with a minimum distance of separation between child nodes in individual clusters, and DAG inter-cluster parent nodes thereby reducing the overall area.

3.3 Node Interleaving

In an individual DAG cluster with finite lengths of string labels that are usually horizontal, graph orientation necessitates consideration of node heights instead of node widths in the top and bottom regions. If the nodes are vertically labeled, the approach can be easily adapted. *Node interleaving* is performed in these regions to completely eliminate node overlap. The node space of each DAG cluster is divided into four quadrants Q_1, Q_2, Q_3, and Q_4 as depicted in Figure 2.

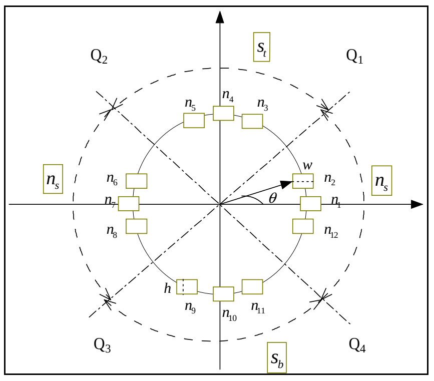


Figure 2. Partitioning of node space - quadrants and sectors.

The top sector (s_t) contains child leaf nodes within a 90 degree zone in quadrants Q_1 and Q_2. Similarly, the bottom sector (s_b) contains child leaf nodes within a 90 degree zone in quadrants Q_3 and Q_4. The remaining nodes that do not belong to either of these sectors are considered as non-sector nodes (n_s) and defined by:

$$N_{ns} = N_{tot} - (N_{st} + N_{sb}) \qquad (4)$$

where N_{ns} = total number of non sector nodes in the DAG cluster, N_{tot} = total number of child leaf nodes in the DAG cluster, N_{st} = total number of top sector nodes in the cluster, N_{sb} = total number of bottom sector nodes in the cluster. In order to achieve node interleaving for sectors s_t and s_b, we divide each sector into odd and even numbered child leaf nodes. Additional tolerances for odd and even child leaf nodes in the top and bottom sectors in a DAG cluster result in non-uniform edge lengths and removes overlap. We consider node heights (n_h) instead of node widths (n_w) for calculating edge lengths in Formula 2.

Hence average edge length (l_e) of individual DAG clusters for achieving minimum distance of node separation is given by:

$$l_e = \left(\left(\frac{n_w + \delta}{2\sin\left(\frac{180^\circ}{N_{ns}}\right)} \right) * N_{ns} + \left(\left(\frac{n_h + \delta}{2\sin\left(\frac{180^\circ}{N_{se}}\right)} \right) + \delta_{se} \right) * N_{se} + \left(\left(\frac{n_h + \delta}{2\sin\left(\frac{180^\circ}{N_{so}}\right)} \right) + \delta_{so} \right) * N_{so} \right) \Big/ (N_{ns} + N_{se} + N_{so}) \qquad (5)$$

where, n_w = node width, δ = overall edge length tolerance, n_h = node height, N_{ns}= total number of non sector child leaf nodes, δ_{se}= additional tolerance for even nodes in top, bottom sectors, N_{se}= total number of child leaf even nodes in top, bottom sectors, δ_{so}= additional tolerance for even nodes in top, bottom sectors, N_{so}= total number of child leaf odd nodes in top, bottom sectors. So far, we have considered DAGs where child leaf nodes in a cluster have uniform label length. For DAG clusters where child leaf nodes have non-uniform string label length, the average edge length employing node interleaving is given by:

$$l_e = \left(\left(\frac{n_{wa,ns} + \delta}{2\sin\left(\frac{180^\circ}{N_{ns}}\right)} \right) * N_{ns} + \left(\left(\frac{n_{ha,se} + \delta}{2\sin\left(\frac{180^\circ}{N_{se}}\right)} \right) + \delta_{se} \right) * N_{se} + \left(\left(\frac{n_{ha,so} + \delta}{2\sin\left(\frac{180^\circ}{N_{so}}\right)} \right) + \delta_{so} \right) * N_{so} \right) \Big/ (N_{ns} + N_{se} + N_{so}) \qquad (6)$$

where averages $n_{wa,ns}$= average node width of non-sector child leaf nodes $\frac{\sum_{i=1}^{N_{ns}} n_{wi}}{N_{ns}}$ for (i=1,2,…, N_{ns}), $n_{ha,se}$= average node height of even top, bottom sector child leaf nodes $\frac{\sum_{j=1}^{N_{se}} n_{hj}}{N_{se}}$ for (j=1,2,…, N_{se}), $n_{ha,so}$= average node height of odd top, bottom sector child leaf nodes $\frac{\sum_{k=1}^{N_{so}} n_{hk}}{N_{so}}$ for (k=1,2,…, N_{so}), rest of the terminology follows Formula 5. Formulas 3 and 5 determine the stop condition for DAG expansion employing node interleaving for graphs with uniform label length nodes, while Formulas 3 and 6 determine the stop condition for graphs with non-uniform label length nodes.

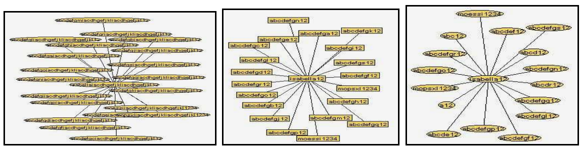
4. EXPERIMENTAL RESULTS

We have implemented the node interleaving enhancements for *E-Spring* algorithm presented in Section 3 for directed acyclic graphs. The results are presented in this section, with comparison of the results with and without node interleaving, for eBay data and graph drawing benchmark data. For the clarity of presenting the experimental results, we use terminology *ESpring_stop* for stop condition in Section 3.1, and *ESpring_interleave* for optimization in Section 3.2. In this implementation, DAG $G_D = (V,E)$ has the following parameters:

- Number of DAG clusters $1 \leq |C| \leq 10$
- Total number of nodes in DAG $1 \leq |N| \leq 160$
- Number of child leaf nodes in DAG cluster $1 \leq n_j \leq 25$

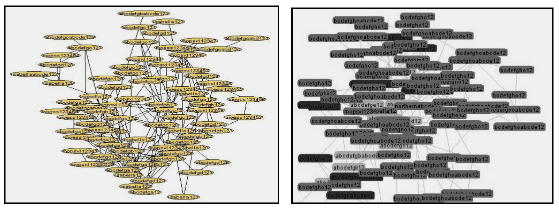
The input data for this implementation is eBay buyer/seller relationships obtained from eBay [15], where user-name string length '*S*' is chosen from one of the following three cases: i) S = 3 ii) S = 10, and iii) S = non-uniform (3 ≤ S ≤ 10). Figure 3a shows the screenshot for S = 30 oval child leaf nodes, Figure 3b represents rectangular child leaf nodes for S = 10, and Figure 3c depicts screenshot for oval child leaf nodes with S = non-uniform label length (3 ≤ S ≤ 10). As an added validation, we benchmarked GraphML data file "g_100_10.xml" which contained the maximum number of nodes N = 100, downloaded for a random DAG from the graph drawing Web site [26], and label string length 10 ≤ S ≤ 15. This DAG does not follow our standard cluster partition into parent and child leaf nodes, however it was interesting to observe that an improvement of 48% in overall area was obtained in this scenario, though a few overlaps occur, as compared to output obtained from Prefuse visualization toolkit [27]. Figure 4a is a screenshot for E-Spring with the stop condition and Figure 4b for *Prefuse*. Figures 5a, b depict resultant graphs obtained after applying stop condition (*ESpring_stop*) and node interleaving (*ESpring_interleave*) for |C| = 10, S = 3 respectively. Figures 6a, b show the resultant graphs obtained after applying stop condition and node interleaving for |C| = 10, S = 10 respectively.

The plotted curves for overall areas of the bounding box are shown in Figure 7a, and % improvements in Figure 7b. Figures 7a and 7b indicate that the percentage improvement of overall graph areas varies between 45% and 79% for *ESpring_stop* and *ESpring_interleave* as compared to *E-Spring*. For DAGs with smaller number of clusters (|C| = 1, 2, 3) and smaller label lengths (S = 3), the variation in the area improvements is small as compared to larger number of clusters (|C| = 10) and larger label lengths (S = 10) in both cases. This proves the usefulness of our approach in effectively visualizing graphs with long and non-uniform node labels. We also observe the improved aesthetic properties like node overlap removal, readability and overall graph symmetry in the resultant graphs obtained after applying node interleaving. This is true for graphs with nodes of uniform string label length (S = 3, 10, 20, 30), non-uniform (1 ≤ S ≤ 10) string label lengths, varying node shapes (oval, rectangular) and varying numbers of clusters (1 ≤ |C| ≤ 10).



a) b) c)

Figure 3. Graphs for |C| = 1, S = 30 (a), rectangular (b), non-uniform (c) nodes.



a) b)

Figure 4. Benchmarking ESpring_Stop (a) interleave (b).

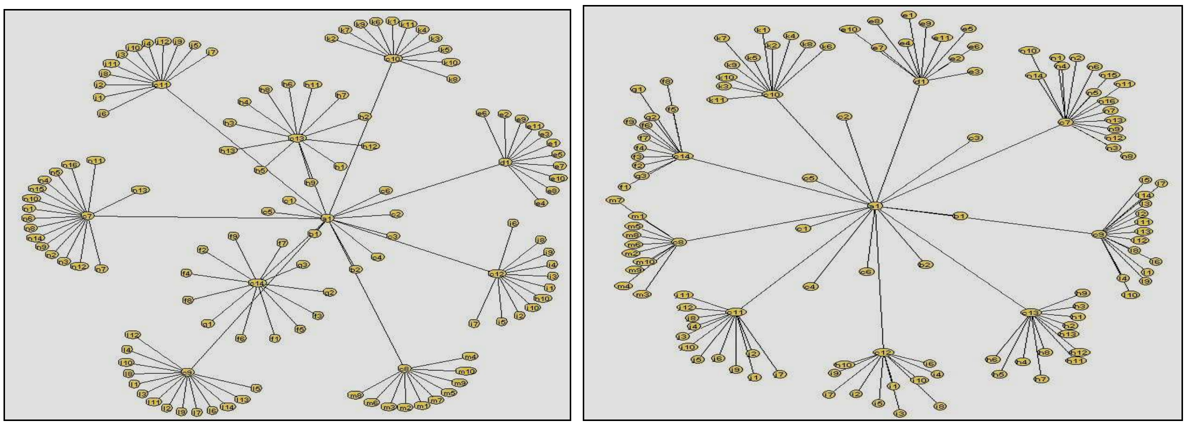

a) b)

Figure 5. ESpring_stop (a) and ESpring_interleave (b): S = 3, |C| = 10.

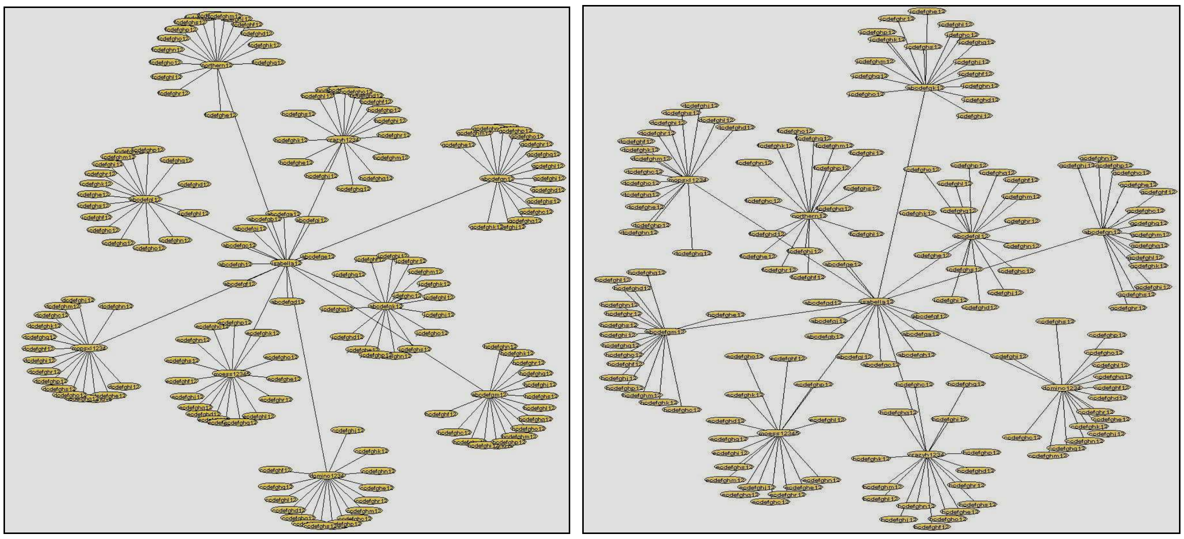
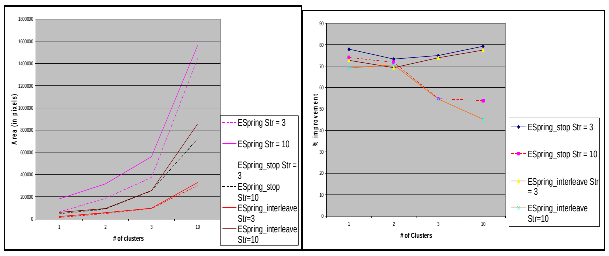

a) b)

Figure 6. ESpring_stop (a) and ESpring_interleave (b): S = 10, |C| = 10.



a) b)

Figure 7. Plots for Area sizes (a), Area improvements (b).

5. CONCLUSION

Visual space is often at a premium and thus reducing the overall area of the bounding box of a graph is desirable. E-Spring algorithm was proposed to draw clustered directed acyclic graphs (DAGs), and uses physical properties of electric charges and spring constants to remove overlaps between node labels. To stop the animated drawing of graphs, we propose a *stop condition* that significantly reduces the overall graph area while at the same time eliminates overlaps. It imposes an upper bound on the graph expansion based on the average edge length of individual clusters and inter-cluster distances. The reduction of average edge lengths within individual clusters and inter cluster distances aid in the minimization of overall graph area and removal of node overlap. Automatic detection of stopping condition bounds the expansion of graphs. Further optimization is achieved using *node interleaving* that performs node interleaving in some sectors of individual clusters to remove residual overlap present in these regions. Implementation on DAGs of varying sized clusters with nodes of uniform label length, non-uniform label length, oval and rectangular shapes as well as benchmarking data with random DAGs was performed; and our experimental visualization results are promising. The overall graph area reduction achieved using enhancements to E-Spring algorithm varies between 45% to 79%, depending on the string length and number of clusters. Future work includes application of this modified algorithm for more graph topologies, scaling to large DAGs, and application graphs such as class diagrams in UML.

6. REFERENCES

[1] Purchase, H. C., Cohen, R. F., and James, M. Validating Graph Drawing Aesthetics. In *Proceedings of the Symposium on Graph Drawing (GD '95)* (Passau, Germany, September 20-22, 1995). Springer-Verlag London, UK, 1995, 435–446.

[2] Kaufmann, M., and Dorothea, W. Drawing graphs, methods and models. Springer-Verlag, 2001.

[3] Battista, G.D., Eades, P., Tamassia, R., and Tollis, I.G. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.

[4] Li, W., Eades, P., and Nikolov, N. Using spring algorithms to remove node overlapping. In Proceedings of the 2005 Asia-Pacific symposium on Information visualization (APVIS '05) (Sydney, Australia, January 27-29, 2005). Australian Computer Society Inc., Darlinghurst, Australia, 2005, 131-140.

[5] Quigley, A., and Eades, P. FADE: Graph Drawing, Clustering, and Visual Abstraction. In Proceedings of the 8th International Symposium on Graph Drawing (GD '00) (Colonial Williamsburg, VA, USA, September 20-23, 2000). Springer-Verlag London, UK, 2000, 197-210.

[6] Huang, X., Lai, W., Sajeev, A.S.M., and Gao J. A new algorithm for removing node overlapping in graph visualization. *Information Sciences An International Journal, 177*, 14 (July 2007), 2821-2844.

[7] Eades, P., and L. Xuemin. Spring Algorithms and Symmetry. In Proceedings of the third International Conference on Computing and Combinatorics (COCOON '97) (Shanghai, China, August 20-22, 1997). Springer-Verlag London, UK, 1997, 202-211.

[8] Kamada T., and Kawai, S. An algorithm for drawing general undirected graphs. *Information Processing Letters, 31,* 1 (April 1989), 7–15.

[9] Kumar, P., Zhang, K., and Wang, Y. Visualization of Clustered Directed Acyclic Graphs without Node Overlapping. In *Proceedings of the 12th International Conference on Information Visualization (IV '08)* (London, UK, July 9-11 2008). IEEE Computer Society, Washington, DC, USA, 2008, 38–43.

[10]Leymann F. *Web Services Flow Language (WSFL 1.0),* IBM, May 2001.

[11]Korb, K. B., and Nicholson, A. E.. *Bayesian Artificial Intelligence*. Chapman & Hall, London, UK, 2004.

[12]Eades P. A heuristic for graph drawing. *Congressus Numerantium*, 42 (1984), 149–160.

[13]Taylor, M., and Rodgers, P. Applying Graphical Design Techniques to Graph Visualization. In *Proceedings of the Ninth International Conference on Information Visualization (IV '05)* (London, UK, 6-8 July 2005). IEEE Computer Society Washington, DC, USA, 2005, 651-656.

[14]Nascimento, H.A.D., and Eades P. User Hints for map labeling. *Journal of Visual Languages and Computing, 19,* 1 (February 2008), 39-74.

[15]Kumar, P., and Zhang, K. Social Network Analysis of Online Marketplaces. In *Proceedings of IEEE InternationalConference on e-Business Engineering (ICEBE '07)* (Hong Kong, China, 24-26 October 2007). IEEE Press, 2007, 363-367.

[16]Lai, W. Layout Adjustment and Boundary Detection for a Diagram. In *Proceedings of Computer Graphics International (CGI '01)* (Hong Kong, China, 3-6 July 2001). IEEE Computer Society, Washington, DC, USA, 2001, 351-354.

[17]Yeh, C., and Gu, Y.Y. Technique to minimise area overhead for delay driven clustering. *IEE Proceedings of Computers and Digital Techniques, 142,* 6 (November 1995), 401-406.

[18]Herman, I., Marshall, S.M., and Melancon, G., Density Functions for Visual Attributes and Effective Partitioning in Graph Visualization. In *Proceedings IEEE Symposium on Information Visualization (INFOVIS'00)* (Salt Lake City, Utah, USA, 9-10 October 2000). IEEE Computer Society Washington, DC, USA, 2000, 49.

[19]Lux, M., Klieber, W., and Granitzer, M. Caliph & Emir: Semantics in Multimedia Retrieval and Annotation.In *Proceedings for the 19th International CODATA Conference The Information Society: New Horizons For Science* (Berlin, Germany, 7-10 November 2004). International Council for Science : Committee on Data for Science and Technology, 2004, 3-7.

[20] Efrat, A., Kobourov, S., Stepp, M., and Wenk, C. Growing fat graphs. In *Proceedings of the eighteenth annual symposium on Computational geometry (SoCG '02)* (Barcelona, Spain, June 5-7, 2002). ACM New York, NY, USA, 2002, 277-278.

[21] Li, W., Eades, P., and Hong, S.H. Navigating software architectures with constant visual complexity. In P*roceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC '05)* (Dallas, Texas, USA, 21-24 September 2005). IEEE Press, 2005, 225-232.

[22] Furnas, G.W. Generalized Fisheye Views. In *Proceedings of Human Factors in Computing Systems (CHI'86)* (Boston MA, April 1986). ACM Press, 1986, 16-23.

[23] Eades, P., and Feng, Q.W. Multilevel Visualization of Clustered Graphs. In *Proceedings of Symposium on Graph Drawing (GD '96)* (Berkeley, California, USA, September 18-20). Springer-Verlag, Berlin, 1997, 101-112.

[24] Herman, I., Marshall, M.S., and Melançon G. Graph Visualisation and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics, 6,* 1 (Jan-Mar 2000), 24-43.

[25] Eades, P., Lai, W., Misue, K., and Sugiyama, K. Layout adjustment and the mental map. *Journal of Visual Languages & Computing,* 6 (1995), 83–210.

[26] URL: *http://www.graphdrawing.org*

[27] URL: *http://www.prefuse.org*