ASSET Series B (2006) 5 (1): 67 -78

ASSET
An
International
Journal

A VISUALIZATION APPROACH TO GRAPH LAYOUT PROBLEM

*A. P. ADEWOLE AND **A. B SOFOLUWE

*Department of Mathematical Sciences, University of Agriculture, Abeokuta, Nigeria **Department of Computer Science ,University of lagos, Akoka, Nigeria.

ABSTRACT

A comprehensive assessment of existing graph layout algorithms revealed that most of the existing algorithms employed automatic approach which permits graph layout without user intervention. However, automatic approach poses some difficulties: (1) user or application-specific layout constraints are usually not taken into account. (2) automatic approach seldom makes use of information in the current layout for the next incremental update. (3) fixed node size is commonly being used. To overcome these difficulties, an interactive graph layout system, **A Visualization Approach To Graph Layout Problem, AVATGLP** based on modified force-directed algorithm was developed. The system revealed that a user or application-specific constraint may be easily added to many automatic graph layout algorithms, and that using information in the current layout incremental update coupled with variable node size, leads to a better graph layout. This study will aid researchers, managers, e.t.c. in gaining new insights and ideas about the relationships inherent in their data-set, and is, therefore, very useful for building sophisticated research tools.

Keywords: Animation , force-directed, graph, layout, visualization.

INTRODUCTION

According to (Battisa et al.(1989) Nelson et al.(2004) and Susan (1996) graph layout is considered to be a challenging problem. It increases in difficulty as the size and complexity of the data increases, and can be particularly onerous if one wants to incorporate readability considerations into the layout (Battisa et al.,1999). To add to this problem, some data visualizations further constrain the layout to reflect data semantics e.g., relative node positioning in relation to the rest of the data.

Presently, automatic layout approach is commonly used when displaying graphs because they provide a drawing of the graph without user intervention. There are, however, several disadvantages to automatic layout. User-or application-specific layout constraints are usually not taken into account. A second problem is that automatic layout algorithms seldom make use of information in the current layout for the next incremental update. This can be frustrating to theuser because whenever a new layout is done, the user's orientation in

the graph is lost. Also, fixed node size is highly encouraged in automatic layout approach.
The investigators concern in this work is how nodes appear within a graph, previous studies have shown that node placement can cause considerable difficulty in how people read graphs (Andrea,1994; Harmanani et al.,2002). Therefore, a visualization approach to graph layout problems using a modified force directed algorithm to solve the above-mentioned problems was developed. The paper showed how user-specified layout constraints might be easily added to many automatic graph layout algorithms.

This approach provide a continuum between manual and automatic layout by allowing the user to specify how stable the graph's layout should be.

RELATED WORK

There have been numerous algorithms proposed to solve the graph layout problem. Force-Directed layout techniques also known as Spring Embedders, according to (Eades, 1983,. 1984; Battista et al., 1999) are a reliable general purpose tool for graph layout applications.

Eades modelled a graph as a physical system of rings and springs, but his implementation did not reflect Hooke's law, rather, he chose his own formula for the forces exerted by the springs. Another important deviation from the physical reality is the application of the forces: repulsive forces are calculated between every pair of vertices, but attractive forces are calculated only between neighbours.

Battista et al. (1999) have their own variant on Eades' algorithm. The authors also modeled a graph as system of springs, but whereas Eades abandoned Hooke's law (Battista et al.,1999) solved partial differential equations based on it to optimize layout. The drawback is that only one vertex moves at each iteration. The repositioning of vertices is repeated until the energy goes below a preset threshold.

Harmanani et al. (2002) tried to be flexible in meeting different aesthetic standards by using simulated annealing algorithm for solving graph layout problem. Simulated annealing is however extremely slow and impractical for interactive display of graphs.

In this study, AVATGLP system based on modified force-directed algorithm, which can reliably provide a satisfactory layout when presented with an unknown graph, was developed.

THE APPROACH TO AVATGLP

The model "Visualization Approach To Graphlayout Problem" AVATGLP shown in Fig.1 is a modified force-directed method based graph layout system consisting of the pressure directed part and force-directed part. This system permits interactive modification of constraints defined among graphic objects (nodes). With an input data representation and constraints defined among components, AVATGLP lay out those components in a rectangle using a modified force-directed algorithm and displays the best solution. After looking at the displayed solution, users of AVATGLP will be able to edit it with graphic editor to modify or add new constraints. For example, when a user notices that an object is placed in a wrong position, he can drag it to the right position and make it immovable to make the whole structure looks neat. Attempts

to develop AVATGLP have proceeded in two main directions: the global layout phase and the post-processing phase.

The Global Layout Phase

This is the phase that determines the size of the placement area for the graph. The size of each of the nodes to be placed on the placement area is also calculated by taking into consideration the image aspect ratio of each of the node and the links (edges) between node pairs are also determined. For instance, given a graph G = (N,E) consisting of a finite set N of nodes and a finite set E of edges(n_i,n_j). The layout of a graph means the placement of this graph on a given area A_p according to certain constraints. Some of these constraints are that:

* the relationship among nodes should be maintained
* the number of edge crossings should be minimized
* user specified constraint should be taken care of
* there should be even distribution of nodes within the placement area.

Unlike what entails in most of the existing automatic approaches where fixed node size is being used, in AVATGLP variable node size is adopted and this has led to effective utilization of placement area since unnecessary wastages in space usage are avoided. The pressure-directed part of AVATGLP is responsible for computing variable node sizes and this is done by applying pressures to each of the node objects. The detail explanation of this approach is discussed in the following sub-section.

PRESSURE-DIRECTED PART

The variable node size is introduced by the pressure-directed part of the AVATGLP model shown in Fig.1. To achieve this, two attributes of nodes which model the tendency of nodes to reach a certain size are used, they are the inner and the outer pressure. The pressure-directed part of this approach is based on Boyle's law.

In this approach, a fixed node size is assumed for all node labels and the presence of equal inner pressure within the rectangle and outer pressure outside the rectangle are also assumed at the initial stage. An incoming token increases the outer pressure which led to a state of imbalance and this eventually led to the increase in the size of the rectangle meant for node labels. This continues until all nodes are placed on the placement area.

This is expressed in pseudocode form as follows:

```
Procedure: Global Layout Phase 3
determine initial placement
determine change in pressure for each node
initialize counter i=0
While I < Node_No -1
{
determine node j with maximum change
in pressure
if change in pressure>0 then
reduce node j in size
else
   enlarge node j
adjust pressure differences
end
i=i+1
}
```

Post-Processing Phase

The result of the global layout phase is a layout that is characterized by overlapping, instability, and inefficient use of the placement area. In this phase, after the nodes dimensions are determined, the placement of the nodes is adapted in such a way that overlapping is

minimized. This phase preserves the stability of the layout as much as possible and it uses the previous layout as an initial solution when the layout algorithm is performed a second time on a modified graph.

The method used in achieving the above mentioned results is based on the force-directed part of the model. The principle of the force-directed method is to use physical simulations to lay out a graph. Forces are calculated, applied to vertices, and recalculated over many iterations. The fundamental function of force-directed methods is to find a layout for the vertices. Vertices are the only objects with forces, directions, or ultimately something equivalent to mass while edges merely act as forces between vertices which will affect their final positions. Each round of the force simulation consists of three main steps: computation of the values of the zone of each node, and the move of each node, with the amplitude bounded by its zone.

Force Directed Part

As the distance between two nodes can become very small in the pressure-directed part of the global layout phase, more iterations in the force-directed part of the algorithm are performed in order to compensate this undesired behaviour. The algorithm stops if the maximum number of iterations is reached or if the improvement of the force differences is too little. An essential advantage of ?

the force-directed part with respect to graph layout problem is that it only performs minor changes of the layout in subsequent iterations. This allows the graphical presentation of temporary layouts and leads to the impression of a continuous flow of the screen objects for the user.

In this study , the following spring forces were used:

*the attracting force F^a

* the repulsive force F^r between each pair of nodes ,and

* the repulsive border force F^{rb} between each node and the borders of the placement area in each direction (+x,-x,+y,-y).

The approach involves modeling pairwise repulsive forces F^r between unconnected nodes and attractive forces F^a between nodes connected by edges. The functional forms of F^a and F^r are arbitrary and the layout represents an equilibrium solution obtained by iterative solution of the corresponding equations of motion.

The functional form of F^a is based on Hooke's law (Eqn.1) which gives the attractive restoring force for elastic springs of natural springs of natural length l_x when stretched by displacement x.

$$F^a = -k^a(x - l_x) \quad \ldots\ldots\ldots\ldots\ldots \quad (1)$$

Real springs are elastic only for small displacements but this attribute can safely be relaxed. In Force-directed algorithm the elasticity of springs due to small displacements is relaxed by setting x to be negative when $l_x > 0$ in order to prevent further compression. This approach is used in this study. The stiffness of the spring is represented as constant K^a and the stiffer the spring, the greater the force required to produce a given extension. Homogenous values are usually assumed for K^a and l_x so that each spring has the same stiffness and each has the same natural length

The functional form for the repulsive fore F^r between nodes is commonly based on an inverse square law behaviour where the strength of the repulsive force between nodes i and j falls off in proportion to the square of the distance between them. This is shown in Eqn. (2).

$$F^r_{ij} = k^r / r^2_{ij} \quad \ldots\ldots\ldots\ldots\ldots \quad (2)$$

In actual practice, some slight modification is needed in order to prevent singularities where the distance, r between nodes becomes very small. It is common to assume homogeneous values for K^r, the constant which determines the strength of the repulsion. Thus, each node exerts a repulsive force on each other node which depends only on their separation. Combining Eqns.(1) and (2) leads to Eqn.(3) for the total force on node i, obtained by combining the individual contributions from individual pairs of nodes as described in Eqns. (1) and (2). The force and distance quantities are now represented by vectors and $\emptyset_{ij}$ denotes the unit (direction) vector between nodes i and j.

$$F_i = \sum_{i,j \hat{a} V' i \neq j} \frac{k^r \emptyset_{ij}}{|r_j - r_i|^2} + \sum_{i,j \hat{a} E' i \neq j} -k^a \times \acute{y} \times \acute{y} r_j - r_i ? \acute{y} \; -l_x \times \acute{y} r_j - r_i ? \acute{y} ? \acute{y} \; |r_j - r_i| \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Equation(3) involves identically charged particles repelling each other while some pairs of particles are joined by springs of identical natural length and stiffness.
However, it was noticed that the particles (nodes) may carry different amount of charge. Therefore, the repulsive force between a pair of nodes will depend on the product of their charges. Similarly, the network of springs connecting nodes may contain some very stiff springs as well as weaker ones and they may have a variety of natural lengths.

In the physical world from which the inverse square between behaviour has been borrowed, the repulsive force would correspond to that between charged particles which might carry different charges as described by Eqn.(4).

$$F^r = 1/4 \text{"} \mathring{a}_0 (q_i q_j / r^2_{ij}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

where: $q_i q_j$ represent the (possibly different) charges on particles (nodes) i and j, respectively and $1/4 \text{"} \mathring{a}_0$ is constant over the entire system.

This study therefore relax the homogeneity aspect of Eqn.(2) by representing the repulsive force between nodes as shown in Eqn. (5).

$$F^r_{ij} = k^r q_i q_j / r^2_{ij} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

The modified version of the force-directed equation (3) is equation (6).

$$F_i = \sum_{i,j \hat{a} V' i \neq j} \frac{k^r q_i q_j \emptyset_{ij}}{|r_j - r_i|^2} + \sum_{i,j \hat{a} E' i \neq j} -k^a_{ij} \times \acute{y} \times \acute{y} r_j - r_i ? \acute{y} \; -l_{xij} \times \acute{y} r_j - r_i ? \acute{y} ? \acute{y} \; |r_j - r_i| \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

The calculation of the repulsive forces F^r and F^{rb} is done in such a way that the force tends towards infinity with decreasing distance. Therefore, an overlapping of objects (nodes) or the placement of an object outside the placement area is minimized.
The pseudo-code for the algorithm is given below:
Area = W * H /W and H are the width and height of the frame/
G = (V,E) /the vertices are assigned random initial positions/

```
For (i = 1;i<iterations;i++) {
/calculate attractive forces/
for e in E {
Δ=e.v.loc – e.u.loc;
e.v.disp = e.v.disp -    Δ  ;
e.u.disp=e.u.disp+  Δ   ;
}
while e in E {
v.loc = v.loc+ (v.disp//v.disp/);
v.loc.x = min(W/2, max(-W/2,v.loc.x));
v.loc.y = min(H/2, max(-H/2, V.loc.y));
}
/Calculate repulsive forces/
  for v in V {
    /each node has two vectors: loc and disp/
      v.disp=0;
       for u in V {
        if (u ≠ v) {
    Δ  = v.loc – u.loc;
   v.disp =  v.disp + v; }}
}}
```

In this study, the investigators do not normally encounter "real" charges or stiffness. In general, the paper deals with graphs G = (N, E) whose nodes n_i and edges e_{ij} each have properties or attributes of their own. Therefore, spring-based forces between each pair of nodes i and j act to improve inter-node separation in the layout.

ANIMATION

In this approach the entire post-processing part is animated. Every change whether triggered by the user or by another agent is animated. The animation is expected to reduce the cognitive effort of the user in recognizing change; our aim is to preserve the user's "mental map". A straightforward approach to generate animations is used. After the new drawing is computed, the nodes will be moved from their initial positions to other positions until they get to their final positions. To accommodate for the human preference of natural motion, the node speed was slowly increased in the beginning and decrease it towards the end.

SYSTEM IMPLEMENTATION AND EVALUATION

AVATGLP is developed as an applet and implemented in Java. AVATGLP Program uses four java files. The overall structure of this program is depicted as shown in Fig. 2.

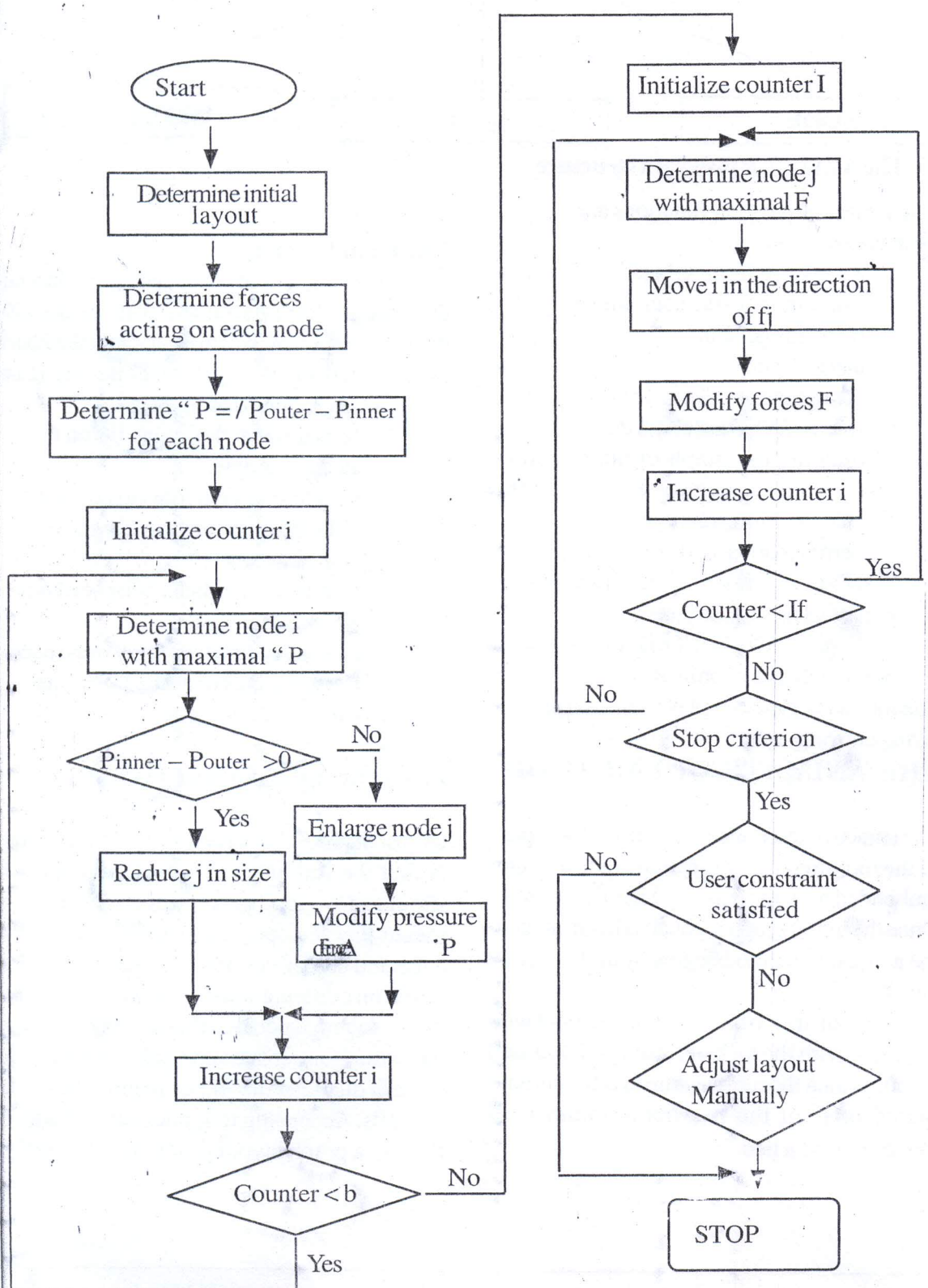

Fig 1 A Visualization Approach To Graph Layout Model

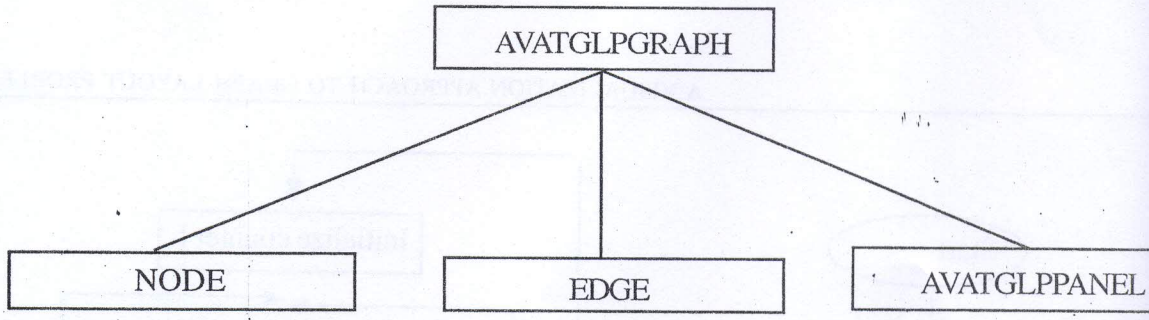

Fig 2. The AVATGLP program structure

.These classes and their functions are summarized following:

- Avatglpgraph Class: This class implements the user interface and performs general initialization.
- Avatglppanel Class: This class assists the AVATGLPGRAPH class by drawing the graph of nodes and by carrying out animation process. It also provides functions that permit manual node movement.
- Node Class: This class stores individual node coordinates.
- Edge Class: This class stores individual edge coordinates.

The following methods play a vital role in laying out the graph:

THE ANIMATION( ) METHOD

The attractive force and the repulsive force part of the modified force-directed algorithm are embedded and implemented in this method. Once the initial graph layout has been created the animation method begins by updating the layout.

The program then begins a while loop that will continue with the task of finding an optimal layout as long the running thread is the current thread or until the maximum number of iterations is reached.

The Paint Method 0

This is the method that is responsible for draw the graph. It draws the nodes, fill the nodes w the appropriate information, and link the no as arranged in the data structure. Its m functions are itemized as follows:

- it ensures that the image fits on the display window.
- it made available fonts to be used
- it fills the placement area with specified color
- it obtains individual node coordinat from node class
- it calculates the length between nod
- it draws edges to nodes positions using nodes addresses.

PERFORMANCE EVALUATION

One of the most important goals with the wo is that the implementation of the approa should show good performance. By this i meant that the approach should achieve t expected objectives. Performance evaluati based on computational time was carried o The result revealed that this approach just li the existing automatic approach ensures that t system displays the initial result within fe seconds. According to Ignacio and Thadde (2004), a graph layout problem is a hard *

User or application-specific layout constraint are taken into account by

providing a graphic editor to modify or add new constraints.

* Information in the current layout are used for the next incremental update in AVATGLP.

* Variable node size is used instead of fixed node size used in an existing automatic approach

In addition certain aesthetic criteria such as: Distributing the vertices evenly in the frame, Conformity to the frame, Minimization of edge crossings, Making edge length uniform, and Reflecting inherent symmetry have been proposed for graph layout evaluation. Although, there have been a lot of arguments as to which of the proposed aesthetic criteria are to be used for judging whether or not a graph layout solution presented is the optimal.

AVATGLP does not explicitly strive for these goals, but does well at distributing vertices evenly, at minimizing edge crossings, and at conforming image placement to the placement area. Automatic approach equally does well in minimizing edge crossings and at conforming image placement to the placement area. However, when considering evenly distribution of vertices AVATGLP performs excellently well as compared to automatic approach whose final output is characterized with node overlapping especially when dealing with fairly large number of nodes.

In order to further estimate the efficiency of our approach for use with graph layout, the authors implemented both the automatic approach that lay out graphs automatically without user intervention and AVATGLP that lay out graphs using visualization approach. The two approaches were subjected to the same treatment using the same set of data items contained in HTM files 1 – 10 as indicated in table 1. The results obtained are shown in table 1 with the graphical comparison shown in Fig 3

Table 1. Performance evaluation of AUTOMATIC and AVATGLP approaches based on node overlapping

Graph	No. Of Nodes	AUTOMATIC	AVATGLP
		No of Overlapping	No of Overlapping
Graph1.html	12	0	0
Graph2.html	33	18	0
Graph3.html	37	6	2
Graph4.html	21	5	0
Graph5.html	41	12	0
Graph6.html	20	3	0
Graph7.html	20	0	0
Graph8.html	27	4	0
Graph9.html	52	2	0
Graph10.html	112	62	33

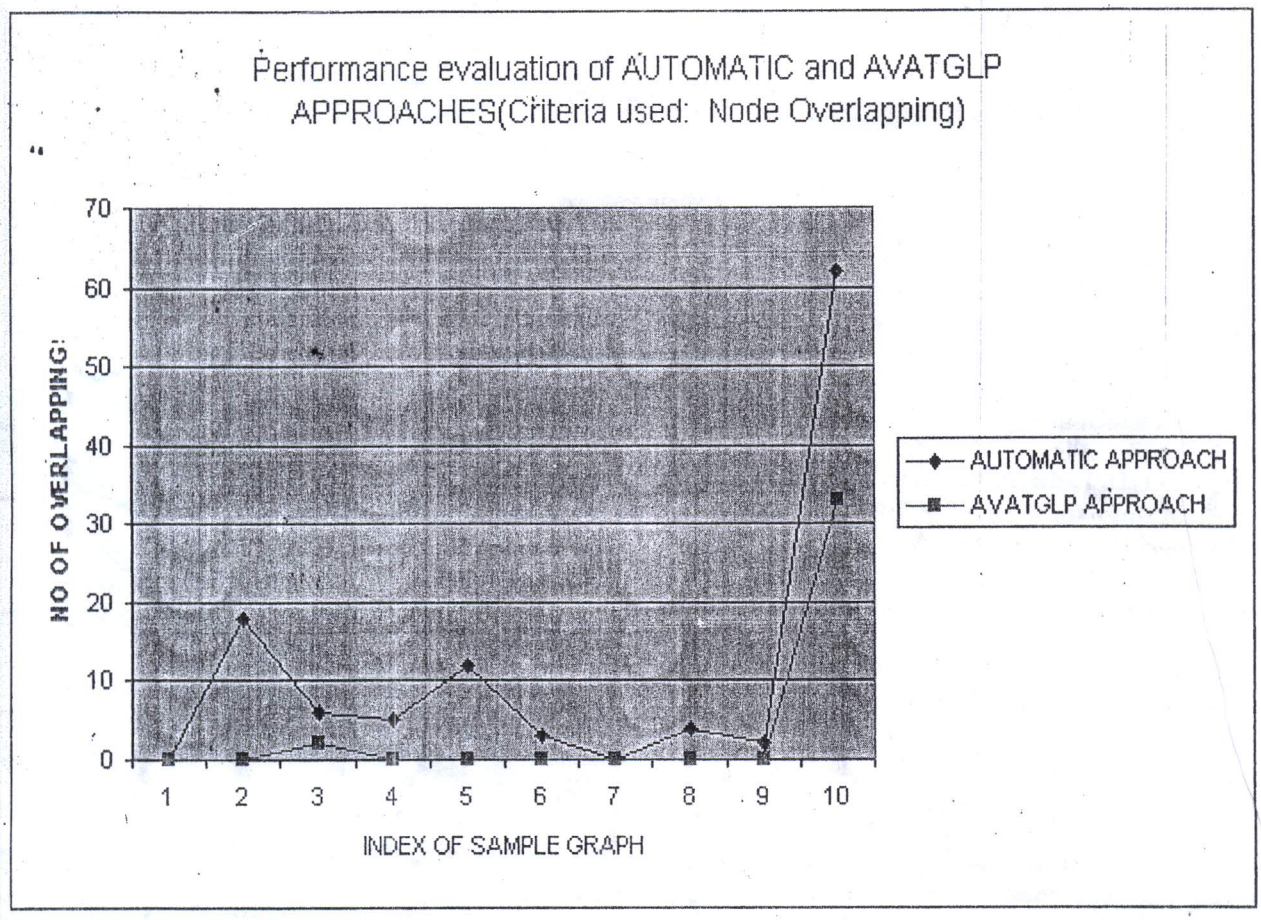

Fig 3. Performance evaluation of AUTOMATIC and AVATGLP approaches based on node overlapping

CONCLUSION

A layout approach has been presented which is suitable for graph layout and it offers significant performance improvement over standard automatic method for force-based placement. The approach uses a visualization approach and force-directed graph drawing algorithms to compute visual representations of graphs depending on the structure of the data. Animations between subsequent drawings enable users to explore different functional layouts without getting lost in the entire graph layout. The approach used in this system has been able to provide solution to the three problems discovered in an existing "Automatic Approach". These problems are handled in AVATGLP as follows:

- User or application-specific layout constraint are taken into account by providing a graphic editor to modify or add new constraints.
- Information in the current layout are used for the next incremental update in AVATGLP.
- Variable node size is used instead of fixed node size used in an existing automatic approach.

This visualization approach has been tested with different set of data based on linked list data structure and the approach offers a user- friendly way to explore graph layout.

REFERENCES

Andrea, G.B.T 1994. Drawing Graphs withEvolutionary Algorithms, *Journal of Algorithms*,15(3), pp.127-139.

Battista, G. D., Eades, P., Tamassia, R., Tollis, I.G.(1989).Algorithms for Drawing Graphs, *An annotatedBibliography, Technical Report*.URL:ftp:Wilma.cs.brown.edu/pub/papers/compego/gdbiblio.ps.z

Battista,G.D., Eades, P., Tamassia, R., Tollis, I.G.1994. "Algorithms for drawing graphs. *An annotated Bibliography", Computational Geometry: Theory and Applications*, Vol.4,5, pp.235-282. ftp.cs.brown.edu.pub/papers/compego/

Battista,G.D., Eades, P., Tamassia, R., Tollis, I.G. 1999. Graph Drawing Algorithms for the Visualization of Graphs, *New Jersey: Prentice Hall*. doi:10.1057/palgrave.jors.2601647.

Blythe, J., McGrah, C., Krackhardt, D. 1995. The Effect of Graph Layout on Inference from Social Network Data, *Proceedings of the Symposium on Graph Drawing, GD'95, Springer-Verlag*, pp.40-51.

Dengler,E.,Cowan,W. 1998. Human Perception of Laid-Out Graphs, *Proceedings of the Symposium on Graph Drawing GD'98, Springer-Verlag*, pp.441-444.

Eades, P., Sugiyama, K. 1990. How to draw a directed graph, *Journal of information processing* Vol.13, No.4, pp.424-434.

Eades, P., Mao, L.H.2000. Navigating Clustered Graphs Using Force-Directed methods. *Journal of Graph Algorithms and Applications*. http://www.cs.brown.edu/publications/jgaa/

Emden, R.G., Stephen, C. N. 2000. Improved Force-Directed Layouts, *In Symposium on Graph Drawing GD'2000,* Vol. 1032 of Lecture Notes in Computer Science, pp.231-240.Ersten, C., Kobomov, S.G., Le, V., Narabi, A. 2004. Simultaneous Graph Drawing Layout Algorithms and Visualization Schemes, *In IEEE Symposium on Information Visualization (InfoVis'04)*, pp1-11.

George, R. M.1987. *Computer Graphics in Application, Prentice – Hall International Edition*, PP.211 – 242.12

Harmanani, H., Zouein, P., ASCE, A.M., Hajar, A. 2002. An Evolutionary Algorithm for Solving the Geometrically Constrained Site Layout Problem, *Journal of Computer in Civ. Engrg.,ASCE,* 16(2), pp.331-338.

Ignacio, C., Thaddeus, S. 2004. A Spring-Embedding Approach for the Facility Layout Problem, *Journal of the Operational Research Society*, 55,pp.73-81.

Martin T., Estrin L. 1987. Models of Computations and systems – Evaluation of vertex probabilities in Graph Models of Computations, *Journal of the ACM*, PP.349 – 366.

Nelson, W., Sheelagh, C., Saul G. 2004. EdgeLens: An Interactive Method for Managing Edge Congestion in Graphs, *Journal of Graph Algorithms and Applications*, 8(1),pp69-74.

Neville, C., Warwick, I., Carl C. 2004. Inhomogeneous Force- Directed Layout Algorithms in the Visualization
Pipeline: From Layouts to Visualization, *Conferences in Research and Practice in Information Technology*, 35,pp.121-128.

Susan, S.(1996). Automatic Graph Drawing Algorithms, *DIMACS International Workshop, Springer Verlag, Berlin*, pp1-15.