A Network Topology Auto-Layout Algorithm
Based on “Concentric-Arrange” Model

Lu Yang
Computer Networks and Distributed Systems Laboratory, Peking University
Room 1716, Science Building No.1
Peking University, Peking, China
+86-10-62751799-8037
yanglu@net.cs.pku.edu.cn

Wei Yan
Computer Networks and Distributed Systems Laboratory, Peking University
Room 1715, Science Building No.1
Peking University, Peking, China
+86-10-62765811
yanwei@cs.pku.edu.cn

Xiaoming Li
Computer Networks and Distributed Systems Laboratory, Peking University
Room 1711, Science Building No.1
Peking University, Peking, China
+86-10-62765811
lxm@cs.pku.edu.cn

ABSTRACT
A new algorithm was advanced in this paper to do the auto-layout of the network topology. All the vertexes and edges of a graph are assigned some geometrical properties such as degrees, outer-degrees, coordinate and consistency. Thus, these vertexes will be arranged around the centre of a circle and from the inner circle to the outer circle forming a series of concentric circles. This kind of arrangement algorithm is provided with the characteristic of perspicuity, aesthetic perception, and making full use of space, when the number of vertex in the topology is extremely large. The algorithm now is well applied in the General-Purpose Network Management System, Peking University.

Categories and Subject Descriptors
C.2.3 [COMPUTER-COMMUNICATION NETWORKS]: Network Operations – Network management.

General Terms

Keywords
Network management, Network topology, Concentric Circles, Auto-Layout, Concentric-Arrange Model.

1. INTRODUCTION
Most network management software are highly based on the map of network topology, that is, to present routers, sub-networks, other components and their relationships entirely in a plane area, from which the operator could perform configurations, watching for malfunction, charging and other operations. Referring to the construction of network topology map, two aspects are concerned: the auto-discovery of network equipment and the auto-layout of the network topology map. Currently, researches on the auto-discovery of network equipment is in-depth and algorithms are well-developed [1, 2], however, researches on the auto-layout of network topology maps and related algorithms are insufficiently, and the capability of the auto-layout of network topology map in many commercial network management software are simple and roughness [3, 4]. Of course, their algorithm applies well when the number of Vertexes is small. As the number of Vertexes in a network grows to 100,200 or much more, the map will become messy, and hard to understand the arrangement and the relationship between vertexes.

Currently, there are some already applied auto-layout algorithms, as in the following:

1. Row-Line auto-layout. That is, simply arrange the vertexes in a matrix and then connect correlative vertexes.
2. Simple-Round auto-layout. That is, simply arrange the vertexes in a series of concentric circles and then connect correlative vertexes.
3. Gravitation auto-layout, which introduced the conception of gravitation. It has the shortage of low speed of constringency and could not support large number of vertexes.

The new algorithm introduced in this paper is expected to achieve the following effect:

1. Reduce the intersects in the edges between vertexes as much as possible.
2. Uniformity of the vertexes layout, that is, to arrange the correlative vertexes together as much as possible.
3. Clarity of the arrangement. The operator will grasp the framework of current managed network topology at a glance.

The following sections in this paper introduce the basic conception and idea of this new algorithm first. After that, give prominence to the advantage of this algorithm by some examples. At last, discussion on some further researches and works will be described.

2. TERMS/NOTATIONS
This section presents the terms and notations used throughout this paper.

Graph: A abstract geometrical map of the current managed network topology.
Connected-Graph: If any two of the vertexes in the graph are connected (whether via other vertexes) the graph is a Connected-Graph.
Ichnography: If a graph can be complication (that is, could be presented in a plane in a form of edge-disjoin) the graph is a Ichnography.
Degree: The degree of a vertex in a graph, in this paper, is defined the number of direct correlated vertexes of that vertex.
Outer-Degree: The Outer-Degree of a vertex in a graph, in this paper, is defined like this: assume that the layout of the
graph’s already done, as for a vertex in a circle, the outer-degree of the vertex is the sum of the degree of the vertexes which are directly correlated to the source vertex and in the next outer concentric circle.

**Central-Angle Degree:** Assume that a vertex arranged in a circle concentric with other circles is \( V_i \), other vertexes arranged in the same circle are from \( V_0 \) to \( V_m \). The Central-Angle Degree of a vertex \( V_j \) in a circle, in this paper, is defined like this:

\[
\frac{\sum_{i=0}^{n} \text{Outer-Degree}(V_j)}{\pi} \times 2\pi
\]

3. AUTO-LAYOUT ALGORITHM

3.1 Ideas of the Algorithm

First, start with the simplest forms of topology graph. Consider the following forms of topology graph:

![Figure 1. The simplest forms of topology](image)

The above figure presents line-topology, circle-topology, star-topology, in turns. When we arrange them on a paper by hand, they’re in that form. If considering in depth, there is some common properties among the three forms of topology:

1. They both have administrative levels. According to the line-topology, the middle vertex is the first level, and two besides it form the second level; according to the circle-topology, all vertexes form the first level; according to the star-topology, the vertex in the center forms the first level while the vertexes around it form the second level. They are clear in levels.

2. All vertexes in a level are arranged in a circle. According to the above figure, the line-topology and the star-topology both have two concentric circles, while the circle-topology has one concentric circle.

3. All vertexes arranged in a circle make use of space of that circle efficiently. Not much congest occurs and do harm to the perspicuity and preventability of the topology graph.

4. Each two of the edges in the topology graph are not intersecting.

Based on the above illumination, consider the following status:

According to an arbitrary topology graph, assume that each vertex in the graph has its own level, and we could arrange it on a corresponding circle, in the just proper position, relative to other vertexes also having the same level and on the same corresponding circle.

Firstly, find out the first-level vertexes of the topology graph, then, select a proper coordinate to a centre of a circle and a radius, in the presentation plane, then arrange the first-level vertexes on the first circle (Circle 0). The circle-space allocation is based on each vertex’s outer-degree.

Secondly, find out the second-level vertexes of the topology graph, based on the first level, then arrange them on the corresponding second level circle which is concentric with the first circle in the presentation plane, also based on each second-level vertex’s outer-degree and the Central-Angle Degree of the direct connected vertex to it in the first concentric circle.

Then repeat the second progress, until all vertexes in the topology graph have been arranged in a concentric circle. Then, line the correlative vertexes in the presentation plane, the topology graph is done.

The most important variable in this algorithm is the Outer-Degree of a vertex and the Central-Angle Degree of a vertex. In that two points in a plane confirm a very line, since the position of the vertexes in the topology graph is confirmed, the edges in the presentation plane are also confirmed. The following sections concentrate on the position computation of each vertex in the topology graph.

3.2 Algorithm Specification

According to an arbitrary unidirectional graph, \( G(V,E) \), the notation “\( V \)” represents the set of the vertexes in the graph, while the notation “\( E \)” represents the set of the edges in the graph. Other definitions are presents as the following:

- The degree of a vertex is presented as \( \text{Degree}(V_j) \).
- The Outer-Degree of a vertex is presented as \( \text{Outer-Degree}(V_j) \).
- The Central-Angle Degree of a vertex is presented as \( \text{Angel-Degree}(V_j) \).
The level of a vertex $V_i$ in the presentation plane is presented as $\text{Level}(V_i)$. Then, the vertex will be arranged on the concentric circle of $\text{Level}(V_i)$.

Every vertexes in a given concentric circle takes some space of that circle, this space is properly scaled by the central angle that the vertex has taken. Counterclockwise, the start angle that the vertex space begins is presented in the notation $\text{Start-Angle}(V_i)$, while the end angle that the vertex space ends is presented in the notation $\text{End-Angle}(V_i)$.

According to a given vertex in a presentation plane (e.g. 500*400 in pixel), the coordinate of a vertex is represented as the notation $(X(V_i), Y(V_i))$.

The radius of first concentric circle is represented in the notation "R", and the radius of the outer concentric circles is of integer times of "R". Example, radius of circle-0 is R, and radius of circle-1 is 2R, circle-2 is 3R, and so forth.

**Stage 1 of the algorithm:**
This stage of the algorithm gets the degree of each vertex and finds the centre of the most inner circle.

1. Input: unidirectional graph $G(V, E)$.
2. For (each vertex $V_i$ in $G$)
   - Compute Degree ($V_i$)
3. End for
4. Max-Degree-Vertex = Index of Max (Degree ($V_i$)).
5. Output: each vertex’s degree and the index of vertex having the maximum degree.

**Stage 2 of the algorithm:**
This stage of the algorithm gets the Outer-Degree of vertexes in the graph. This stage uses recursion functions, to search the graph in the depth-first manner from the vertex represented by the notation “start-point”. As the stage begins, the start_angle variable should be 0 and the end_angle should be $2\pi$.

The variable $\text{Paint\ DegreeRule}$ is defined in the following statement:

$$\text{Paint\ DegreeRule} = \frac{2\pi}{\text{Outer - Degree(Core)}}$$

The notation “Core” in the above statement represents the centre vertex of all the concentric circles.

1. Input: The output of stage 2
2. Clear all the marks of the vertexes marked in stage 2.
3. DFS-Cal (Graph, start-point, level, start_angle, end_angle)
   - Mark (start-point)
   - Start-Angle (start-point) = start_angle
   - End-Angle (start-point) = end_angle
   - For (each neighboring vertex of start-point $V_i$)
     - If Level ($V_i$) == level+1 and $V_i$ is not marked
       - DFS-Cal (Graph, $V_i$, level+1, start_angle, start_angle + $\text{Paint\ DegreeRule} \times \text{Outer - Degree}(V_i)$)
     - End If
   - End For
   - End DFS-Cal
4. Output: The Central-Angle Degree of each vertex in the graph.

**Stage 3 of the algorithm:**
This stage of the algorithm computes the Central-Angle Degree of each vertex in the presentation plane. This stage uses recursion functions, to search the graph in the depth-first manner. As the stage begins, the start_angle variable should be 0 and the end_angle should be $2\pi$.

The variable $\text{Paint\ DegreeRule}$ is defined in the following statement:

$$\text{Paint\ DegreeRule} = \frac{2\pi}{\text{Outer - Degree(Core)}}$$

The notation “Core” in the above statement represents the centre vertex of all the concentric circles.

1. Input: The output of stage 2
2. Clear all the marks of the vertexes marked in stage 2.
3. DFS-Cal (Graph, start-point, level, start_angle, end_angle)
   - Mark (start-point)
   - Start-Angle (start-point) = start_angle
   - End-Angle (start-point) = end_angle
   - For (each neighboring vertex of start-point $V_i$)
     - If Level ($V_i$) == level+1 and $V_i$ is not marked
       - DFS-Cal (Graph, $V_i$, level+1, start_angle, $\text{Paint\ DegreeRule} \times \text{Outer - Degree}(V_i)$)
     - End If
   - End For
   - End DFS-Cal
4. Output: The Central-Angle Degree of each vertex in the graph.

**Stage 4 of the algorithm:**
This stage is the final stage, the coordinate of each vertex in the graph will be computed, and, finally, do the Lining to the
arranged vertexes in the given presentation plane. This stage uses recursion functions, to search the graph in the depth-first manner. The coordinate of the centre vertex of all concentric circles should be assigned in variables fromX and fromY, as this stage begins.

Input: The Output of stage 3.
Clear all the marks of the vertexes marked in stage 3.
Cal-Location (Graph, start-point, level, fromX, fromY)

\[
X (\text{start-point}) = \text{fromX} + \text{level} \times R \times \cos\left(\frac{\text{Start} - \text{Angle}(s) + \text{End} - \text{Angle}(s)}{2}\right)
\]

\[
Y (\text{start-point}) = \text{fromY} + \text{level} \times R \times \sin\left(\frac{\text{Start} - \text{Angle}(s) + \text{End} - \text{Angle}(s)}{2}\right)
\]

(Note: the notation “s” in the above statement is represented for “start-point”.)

For (each neighboring vertex of start-point \(V_i\))
If Level \((V_i) == \text{level}+1\) and \(V_i\) is not marked
Cal-Location (graph, \(V_i\), level+1, X (start-point),
\[Y (\text{start-point}) \])
End If
End For
End Cal-Location
For (each vertex in the graph)
Line the vertex to the direct connected vertex of it.
End For

Output: The coordinate of each vertexes in the graph in a given presentation plane.

This algorithm finishes in this stage.

3.3 Discussion of This Specified Algorithm
The auto-layout algorithm of a given topology graph specified above is a typical of recursion algorithm, which takes three rounds of recursion in order to compute the coordinate of each vertex in a given presentation plane. It works well, and does not have everlasting looping, however, the following peradventures of this algorithm needs to be discussed:

1. Does the algorithm integrate and cover all the possible conditions?

No, it doesn’t, though it covers the most often appeared conditions. Two especial conditions are not included in the above algorithm.

First, all stages specified above are based on the approbatory fact, that is, the given arbitrary topology graph is a Connected-Graph. Unfortunately, not all given topology graphs are connected graphs. How to deal with a given graph that is not a Connected-Graph? It’s easy, however, no need to modify the algorithm specified above. Before stage 1, we could find out all connected embranchments of the given topology graph. For each embranchment, applies the algorithm as if it were a entire topology graph. Meanwhile, divide the give presentation plane to blocks, in which one block contains the presentation of one embranchment. The algorithm of dividing the given presentation plane is beyond the scope of this paper. According to a given rectangle presentation plane, the simple method to divide the rectangle is to cut the rectangle to several small rectangles, from left to right, based on the levels of the embranchment which it will present.

Second, see figure-1, the circle-topology. All six vertexes of the topology have the degree of 2. This will bring seriously problems to stage 1 of the algorithm specified above. Which vertex has the largest degree? Six vertexes all have the largest degree. However, if not concerned carefully, select the vertex that has the largest degree among the six vertexes randomly, the algorithm also works. The final figure of the topology when the algorithm finishes would be like this:

![Figure 2. The above algorithm applied to circle-topology](image)

It shows clearly that the result is not what we expect. The last line between the most left and most right vertexes is hindered behind the presentation plane. The cause for this happened in stage 1, the selection of the vertex which has the largest degree. In fact, sometimes more than one vertex has the largest degree, and they should form a circle on the presentation plane, not just a centre of a circle, a point.

To deal with this condition, it’s also simple, no need to modify the above specified algorithm much. Import a virtual vertex, just an assistant vertex and not truly exist in the topology. Connect the virtual vertex to all the vertexes in stage 1 that have the largest degree, and let the virtual vertex become the central of all concentric circles. The depth-first search in stage 2 should start from this virtual vertex, and all stages after stage 1 works well throughout the algorithm.
When presenting vertexes and edges on the plane, the virtual vertex and virtual edges imported should not be presented on the plane.

2. Does the result of the algorithm have the characteristic of perspicuity and aesthetic perception?
   The hierarchy of the final topology graph is clear. Though the actual topology may not have the hierarchy as the graph presents, the graph presents the topology hierarchically with the intention of perspicuity and aesthetic perception. The following section will show the effect while the number of the vertexes in the graph grows much more.

3. Can this algorithm assure that no edges on the final presentation plane are intersecting?
   The answer is no. This algorithm seeks the trade off between the complexities of the algorithm itself and the perspicuity of the final arranged graph. The complexities of the given arbitrary topology of a network are hard to expect. First, not all the given topologies of a managed network can be presented on a plane in the form of no-edge-intersecting. Only those ones whose abstract graphs are ichnographies could be presented on a plane in the form of no-edge-intersecting. The prove is beyond the scope of the paper. Second, based on the hierarchical presentation of the final graph, the complexion that two edges are intersecting occurs usually between direct connected vertexes that are geographically far away, the frequency of this complexion is not very often.
   So, though this algorithm could not assure that no edges on the final presentation plane are intersecting, it could avoid edge-intersecting in the final presentation graph as much as possible.

4. APPLICATIONS/COMPARING
   The Algorithm specified above has been applied to the General-Purpose Network Management System [6], in Peking University, China. This section introduces some application examples of the algorithm.

4.1 Comparing This Algorithm to the HP Openview NM 6.1
   The tow figures above have shown the different presentations of the same network topology, the first in the General-Purpose Network Management System, Peking University, applying the auto-layout algorithm specified in this paper, while the second in HP Openview NM 6.1 software. It’s obvious that the first presentation has more perspicuity and aesthetic perception than the second one.

4.2 The Topology of Campus Networks
Network Management System, operating in Peking University. The topology map presentation framework part of the System has applied the Topology Auto-Layout Algorithm specified in this paper, and the topology graph is so beautiful and nice!

5. RELATED WORKS

Based on this topology auto-layout algorithm and Network Management System accomplished by Network Management Group in Computer Networks and Distributed Systems Laboratory, Peking University, the New Technology of Networks team has built the General-Purpose Network Management System, based on J2EE platform and weblogic application server. Now the system is in the testing stage and operating in Peking University, managing the campus networks. I’m the member of the New Technology of Networks team and major in the design of topology auto-layout algorithm. Future researches includes the improve of the topology auto-layout algorithm specified in this paper.

Future researches also include the IPV6 Network Management Research and intelligent alarm mechanisms in the Network Management Systems.

6. ACKNOWLEDGMENTS

We grant many thanks to the direction of Professor Wei Yan, Professor Xiaoming Li, and to the supervise of Yuan Cheng, member of the New Technology of Networks.

7. REFERENCES