New method to optimize Force-directed layouts of large graphs

Meva DODO dodo@irit.fr
Fenohery ANDRIAMANAMPISOA fandriam@irit.fr
Patrice TORGUET torguet@irit.fr
Jean Pierre JESSEL jessel@irit.fr

Institut de Recherche en Informatique de Toulouse (IRIT)
118, Route de Narbonne, 31062, Toulouse, France

ABSTRACT

This paper describes a novel method to optimize the force-directed placement algorithm for 3D drawing of large graphs. The main idea behind our approach consists in optimizing the layout by equitably distributing vertices in space. We consider the largest sphere inscribed in the 3D space and the vertices are then assigned random initial positions that are improved by force-directed placement. In order to ensure the effectiveness of the algorithm, we propose a new energy function minimization which uses the conjugated gradient of Fletcher-Reeves. Our algorithm is not only addressed to general undirected graphs but it also produces good layouts for large trees. This work is motivated by our need to offer 3D visualization tools for large computing networks but this first phase will be focused on the graph representation.

Keywords
Algorithm, 3D graph drawing, Force-directed placement.

1. INTRODUCTION

Graph algorithms are used in most applications and tools that visualize complex systems or large databases. Their effectiveness comes from the fact that they highlight the structure of a system by presenting each element as a node and their relations as links. This facilitates the perception process required to understand the structure of complex systems, identify the centre of interest and detect possible anomalies.

Several research projects have been carried out on the graph-drawing algorithms these two last decades. [Bat98a] summarizes the most important works on the graphs that can be classified according to their applicability and required aesthetics.

Generally graph drawing algorithms can be classified in two categories: hierarchical algorithms that distribute nodes according to their hierarchy in the graph, and algorithms based on physical models.

The hierarchical algorithms are often applied on trees and acyclic directed graphs. These algorithms try to identify a suitable pre-tree from the given graph. Thus the vertex set is partitioned in several subsets so that the smallest subset contains the root node and the other nodes are partitioned into branches. Nodes are placed successively from the root to leaf nodes.

In contrast, algorithms based on physical models focus on undirected general graphs. The family of these algorithms, generally called force-directed placement, involves transforming vertices and edges into a system of forces and finding the minimum energy state of the system. This state is found by running a simulation of the forces or by resolving differential equations.

The algorithm that we present in this paper falls into the second category of graph drawing techniques. Indeed, undirected graphs can be considered as the most general class of graphs because they allow the representation of any type of information and are generally used in the representation of complex systems.

Representing visually hundreds or thousands of vertices in a small area however remains a principal challenge so that several techniques have been associated with the force-directed methods in order to reduce the poor running time and to produce optimal drawings.

This work presents new techniques that optimize the layout by distributing the vertices equitably, which therefore increasingly reduce edge crossing. Our algorithm is addressed to 3D graph-drawing and its basic ideas consist in using the largest sphere inscribed in space and recursively distributing this sphere to the various vertices. The final positions of vertices are found after several steps and at each step the position of each vertex is improved by force-directed placement.

The second feature of our algorithm is the implementation of a numerical optimization to compute the final position of each vertex. Indeed, the effectiveness of the force-directed placement algorithm resides in the choice of the energy function and the method to minimize this energy.

The conjugated gradient of Fletcher-Reeves is then implemented to perform the numerical optimization. The conjugated gradient is the most efficient method of optimization [Fle64a], [Gil90a], [Wan05a] and its main advantage is that it minimizes a function with several variables in a relatively short time.

This paper is structured as follows. Section 2 reviews earlier force-directed placement algorithms. Section 3 details the contribution of this work, i.e., the vertices distribution approach, the definition of attractive and repulsive forces and the new energy function. We then describe our force-directed placement algorithm at the end of this section. Section 4 presents initial results and shows the capacity of our algorithm to draw large undirected graphs and trees. Finally we conclude and outline perspectives in section 5.

2. RELATED WORK

Most graph-drawing algorithms are based on the physical model that represents vertices as steel rings and edges as springs. Vertices are initially placed at random positions and the system is released so that springs modify the coordinates of vertices until stability is reached (i.e. when the generated total energy is minimal).

This concept was initially developed by Eades [Ead84a] and several projects have been successively carried out. [Kam89a] improved Eades' work by introducing the concept of ideal distance between vertices that are not neighbors. This distance is proportional to the length of the shortest path between them. They are also the first who formulated the total energy of a graph and found that the searching for the vertices' final positions that minimize edge crossings is a process reducing the total energy of a physical system of rings and springs. [Fru91a] developed a new variant of Eades' algorithm but, instead of solving differential equations to minimize the total energy of the system, they used a simulation of repulsive and attractive forces to find the final position of each vertex. Most recent force-directed algorithms are based on the approach of Fruchterman & Reingold. Although they produce aesthetic results for medium graphs, these algorithms have two major drawbacks. The first is that they do not take into account space optimization. Consequently, the vertices are not effectively distributed in space. The second drawback is time consumption. Indeed, these algorithms are based on the principle that every pair of vertices exerts repulsive forces but also that attractive forces are only calculated between neighbors. Before obtaining the final state of the system, several iterations are computed and the positions of the vertices are modified. Given that each iteration is computed in a time of the order of $O(n^2+m)$, n being the total number of vertices and m the total number of edges, this poses a major problem in drawing complex graphs formed by thousands of vertices.

In order to improve quality of drawing and to reduce the total execution time for large graphs, several methods were combined with force-directed algorithms. [Wal03a] presented a multilevel technique based on the partitioning of a graph into many sub-graphs. Thus the algorithm starts with the construction of the smallest sub-graph, improves the positions of vertices with the force-directed placement and uses the obtained result in the construction of the next sub-graph. Similarly, [Gaj04a] has recently combined a new technique called MIS (Maximal Independent Set) with the force-directed placement to draw large graphs. MIS consists in finding a partition of vertices set in many independent subsets. A subset is independent if no pair of elements is connected by an edge.

The major advantage of these techniques is the reduction of execution time since the force-directed algorithm is only applied to the improvement of vertices' positions in a subset at any given time. We only quote here a few techniques but the idea of partitioning large graphs in many sub-graphs had already been introduced by Fruchterman & Reingold with their multi-grid technique. In the same way the idea of partitioning into multi-layer was also used by [Dav96a] and [Har02a].

Initially, the majority of force-directed placement algorithms are applied in two dimensional layout but several studies have proved that three dimensional layouts are visually better and have several advantages. [Dwy01a] confirmed this result in his study on 3D visualization of UML class diagrams using force-directed placement. [Chu01a] propose virtual worlds which can allow users to comprehend large graphs. They have developed a platform for experimenting with 3D force-directed placement algorithms.

3. OUR ALGORITHM

3.1 Overview

The algorithm that we propose in this paper is addressed to 3D general undirected graph-drawing. We will later show that it is also suited to the drawing of large trees. Our algorithm is based on the approach of Kamada & Kawai that consists in solving differential equations to minimize total energy. Indeed, the process for finding the final positions of vertices by simulating attractive and repulsive forces can rarely minimize total energy. This is why each algorithm based on this method gives its own energy function that is never expressed explicitly. Instead of using the partitioning method like recent algorithms, we are interested rather in optimizing the layout by considering each graph as complete, i.e. each vertex is related to other vertices by n-1 links, n being the total number of vertices. This produces a better representation of less complex graphs. Indeed, the idea of partitioning a graph into independent subsets is not very reliable in the sense that it is not always applicable to complete graphs that are impossible to subdivide into independent subsets. The optimization approach that we propose consists in subdividing the largest sphere inscribed in space into several smaller spheres of the same dimension and containing one vertex. The positions of vertices are improved by force-directed placement in order to minimize the unused space in the initial sphere.

The main features of the algorithm are then:

- an equitable distribution of vertices,
- no collision between vertices,
- a minimization of unused space,
- an effective energy function.

3.2 Vertices distribution approach

The key feature of this approach is to build the largest sphere inscribed in the 3D space and to equitably divide it for the various vertices. Let S_0 be the initial sphere, n vertices are distributed inside according to the following conditions:

- each vertex i is at the centre of the partition S_i, where S_i is a sphere,
- for all vertex i and vertex j, the partition S_i has the same size as the partition S_j, i.e. volume(S_i)=volume(S_j).

For this, we have to maximize the volume of each sphere S_i, i.e. to minimize the unused space. Maximizing the volume of each sphere S_i comes down to finding the final position of each vertex minimizing the total energy of the corresponding spring system.

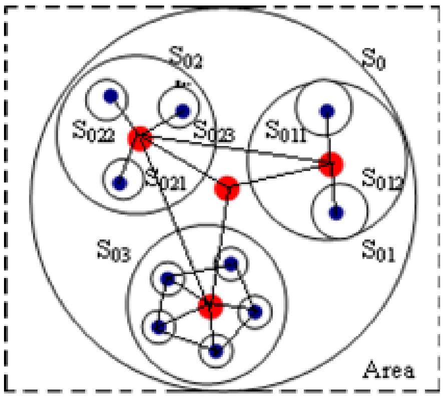

Figure 1: Vertices distribution.

The figure above shows the process of vertices distribution. Colored objects placed in the centre of spheres represent vertices. This is an example of a graph that mixes both a tree and a general undirected graph. Positioning vertices of undirected graphs is easier because the initial sphere is directly subdivided into the various vertices. The only condition to satisfy is that all spheres must have the same volume.

Space optimization is the main interest of the recent algorithm developed by [Ngu02a] for the 2D representation of large hierarchical systems. Their algorithm is very effective because, on a small surface, it lays out thousands of nodes structured in a tree. Unfortunately, it does not apply to topologies structured differently such as complete or general undirected graphs.

3.3 Attractive forces, repulsive forces and total energy

In order to minimize edge crossings and refine the drawing, the principle of force-directed placement is slightly deviated from the physical reality and two forces are introduced: repulsive forces are applied to every pair of vertices and attractive forces that are only applied to adjacent vertices so that they are closer to each other. Thus, the final state of the system is found when the sum of the forces applied is null or minimized down to a chosen value.

Our algorithm is always based on this principle but the main difference resides in the expression of attractive forces. Indeed, in order to optimize the available embedding area and to avoid positioning vertices outside a current sphere, the attractive forces are applied to bring the elements towards the centre. The problem is then modeled as follows:

- the vertices exert repulsive forces towards each other,
- each vertex is attracted by the centre of the sphere (S).

In this way, the repulsive force generated by two vertices is inversely proportional to the distance between them, i.e. the closer to each other two vertices are, the more important repulsive forces they exert. Contrarily, the attractive force is proportional to the distance between a vertex and the centre of the current sphere so it is more important when a vertex is further away from the centre. The expression of the forces applied to a vertex is:

$$\boldsymbol{F}_i = \boldsymbol{F}_i^r + \boldsymbol{F}_i^a \qquad (1)$$

with

$$\boldsymbol{F}_i^r = \sum_{j=1 and\, j\neq i}^{n} \left(Radius - Distance\left(P_i, P_j\right)\right)$$

and

$$\boldsymbol{F}_i^a = n * Distance\left(P_i, Centre\right)$$

n being the total number of vertices, Centre the centre of the sphere (S), $\boldsymbol{F}_i^r$ and $\boldsymbol{F}_i^a$ respectively repulsive and attractive forces applied to a vertex i. P_i and P_j are respectively the positions of vertex i and vertex j. Distance(P_i, P_j) is the Euclidian distance between P_i and P_j.

The final state of the system is found when

$$\sum_{i=1..n} \boldsymbol{F}_i = \sum_{i=1..n} (\boldsymbol{F}_i^r + \boldsymbol{F}_i^a) = 0 \qquad (2)$$

According to the Kamada & Kawai approach on which our algorithm is based, the total energy of the system is expressed by the following sum:

$$\boldsymbol{E} = \frac{1}{2}\sum_{i=1}^{n} \left(\left(\sum_{j=1 and\ j\neq i}^{n} \left(Radius - Distance(P_i, P_j)\right)^2 \right) + n\left(Distance(P_i, Centre)\right)^2 \right) \qquad (3)$$

In terms of xyz-coordinates, the expression (3) is given by the formula (4):

$$\boldsymbol{E} = \frac{1}{2}\sum_{i=1}^{n} \left(\sum_{j=1 and\, j\neq i}^{n} \left(\begin{array}{l} Radius^2 + (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - \\ 2Radius\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \end{array} \right) + n(x_i - cx)^2 + n(y_i - cy)^2 + n(z_i - cz)^2 \right)$$

$P_i(x_i, y_i, z_i)$ and Centre(cx, cy, cz) respectively being the position of vertex i and the centre of the largest sphere inscribed in space.

To find the final state of the system, we need to find the local minima of the energy E at each vertex. This means that we need to take partial derivatives of E and solve them in order to find local minima for each vertex. Let $P_i(x_i, y_i, z_i)$ be the initial position of vertex i. Therefore we need the new position $P'_i(x'_i, y'_i, z'_i)$ which locally minimizes energy E, where $x'_i = x_i + tx_i$, $y'_i = y_i + ty_i$, $z'_i = z_i + tz_i$. In other words, we need the translation $T_i(tx_i, ty_i, tz_i)$ of each vertex i which minimizes energy E. By introducing the expression of (x'_i, y'_i, z'_i) we get the formula (5):

$$\boldsymbol{E} = \frac{1}{2}\sum_{i=1}^{n} \left(\sum_{j=1 and\, j\neq i}^{n} \left(\begin{array}{l} Radius^2 + (x'_i - x'_j)^2 + (y'_i - y'_j)^2 + (z'_i - z'_j)^2 \\ -2Radius\sqrt{(x'_i - x'_j)^2 + (y'_i - y'_j)^2 + (z'_i - z'_j)^2} \end{array} \right) + n(x'_i - cx)^2 + n(y'_i - cy)^2 + n(z'_i - cz)^2 \right)$$

3.4 Minimizing the total energy

The common problem of force-directed placement algorithms is to find an efficient energy function *(cost function*) to reduce the total energy of the system. Thus, we need to find for each vertex the position that stabilizes the system. Our method consists in finding the minimum energy using the conjugated gradient of Fletcher-Reeves optimization method. This method has been already used in a Java-based experimental platform [Dan98a] but repulsion forces are computed in O(n log n) using Barnes-Hut tree-code [Bar86].

The gradient of the sum E (5) applied to a vertex i is given by the following expression:

$$\begin{cases} \dfrac{\partial E}{\partial tx_i} = \displaystyle\sum_{j=1 and\, j\neq i}^{n} \left((X_i - X_j) - Radius \frac{X_i - X_j}{\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}} \right) \\ + n(X_i - X_j) \\ \dfrac{\partial E}{\partial ty_i} = \displaystyle\sum_{j=1 and\, j\neq i}^{n} \left((Y_i - Y_j) - Radius \frac{Y_i - Y_j}{\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}} \right) \\ + n(Y_i - Y_j) \\ \dfrac{\partial E}{\partial tz_i} = \displaystyle\sum_{j=1 and\, j\neq i}^{n} \left((Z_i - Z_j) - Radius \frac{Z_i - Z_j}{\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}} \right) \\ + n(Z_i - Z_j) \end{cases}$$

Where $X_i = x_i + tx_i$, $X_j = x_j + tx_j$, $Y_i = y_i + ty_i$, $Y_j = y_j + ty_j$, $Z_i = z_i + tz_i$, $Z_j = z_j + tz_j$

As we deal with n vertices, the sum E (5) results in 2n non-linear and non-independent equations. Therefore, we find the vertex in the system with the highest energy and move it to a location minimizing its energy, i.e. $\frac{\partial E}{\partial tx_i} = \frac{\partial E}{\partial ty_i} = \frac{\partial E}{\partial tz_i} = 0$. This process is repeated until the local energy for each vertex stops decreasing. This is summarized as follows:

- compute the gradient of E for all tx_1, ty_1, tz_1, tx_2, ty_2, tz_2, …, tx_n, ty_n, tz_n
- compute the norms of translations $\Delta_i = \sqrt{\left(\frac{\partial E}{\partial tx_i}\right)^2 + \left(\frac{\partial E}{\partial ty_i}\right)^2 + \left(\frac{\partial E}{\partial tz_i}\right)^2}$,
- identify the vertex m with the highest norm of translation ($\Delta_m \geq \Delta_i \; i = 1..n$),
- minimize local energy E for this vertex through the conjugated gradient with tx_m, ty_m, tz_m as parameters and keep other vertices at their current positions,
- after minimization, vertex m has $T_m(tx_m, ty_m, tz_m)$ as translation, is moved to its new position $P_m(x_m+tx_m, y_m+ty_m, z_m+tz_m)$, and the conjugated gradients of E for all tx_1, ty_1, tz_1, tx_2, ty_2, tz_2, …, tx_n, ty_n, tz_n are recalculated. These steps are stopped when all norms of translations are lower than a value: $\Delta_i < \varepsilon \text{ for } i = 1..n$, i.e. the final state of the system is found.

In order to improve the aesthetic of the layout, we have introduced in the expression of the energy the graph distance between two vertices which is the shortest path between them in the graph. This allows closer positioning of two neighboring vertices. Let d_{ij} be the graph distance between two vertices O_i and O_j. Therefore, the expression of energy E (5) is given by the following sum (6):

$$E = \frac{1}{2}\left(\sum_{i=1}^{n}\left(\sum_{j=1 \text{ and } j \neq i}^{n}\left(\begin{array}{l}(d_{ij} * Radius)^2 + X_{ij}^2 + Y_{ij}^2 + Z_{ij}^2 \\ -2 * Radius * d_{ij}\sqrt{X_{ij}^2 + Y_{ij}^2 + Z_{ij}^2}\end{array}\right) \atop + n(x_i - cx)^2 + n(y_i - cy)^2 + n(z_i - cz)^2\right)\right)$$

where $X_{ij} = x_i - x_j$, $Y_{ij} = y_i - y_j$, $Z_{ij} = z_i - z_j$.

The algorithm is then summarized by the following pseudo-code:

Minimization algorithm of total energy E

```
Assign random placement of vertices:
initialize ($x_i$, $y_i$, $z_i$), i=1…n
Initialize the translations to zero
($tx_i$=0, $ty_i$=0, $tz_i$=0), i=1...n
Compute the gradients of E for all ($tx_i$,
$ty_i$, $tz_i$) and compute $\Delta_i$ with i=1...n
Find the vertex m with $\Delta_m \geq \Delta_i$, i=1...n
While $\Delta_m > \varepsilon$
   $grad \leftarrow \left(\frac{\partial E}{\partial tx_m}, \frac{\partial E}{\partial ty_m}, \frac{\partial E}{\partial tz_m}\right)$
   Assign the descent direction:
   u($u_x$, $u_y$, $u_z$) $\leftarrow$ -grad
   While $\|u\| > \varepsilon$
      $s \leftarrow \|grad\|^2$
      Compute the optimal step p in the
      direction u, while observing the
      different conditions (applying the
      translation vector, vertex m must
      not be outside the available area
      (S) and it must not be in
      collision with the other vertices)
      Update the translation vector with
      ($tx_m$, $ty_m$, $tz_m$) $\leftarrow$ ($tx_m$+p*$u_x$, $ty_m$+p*$u_y$,
      $tz_m$+p*$u_z$).
      Compute the gradient of E for txm,
      tym, tzm, $grad \leftarrow \left(\frac{\partial E}{\partial tx_m}, \frac{\partial E}{\partial ty_m}, \frac{\partial E}{\partial tz_m}\right)$
      $s \leftarrow \frac{\|grad\|^2}{s}$
      $u(u_x, u_y, u_z) \leftarrow -grad + s * u(u_x, u_y, u_z)$
   End While
   Compute the gradient of E for all
   $tx_i, ty_i, tz_i$ and the $\Delta_i$ with i=1...n
   Find vertex m where $\Delta_m \geq \Delta_i$, i=1...n
End While
```

To compute the optimal step p, it is possible to use minimization by “quadratic interpolation” or by section in “Golden Search” or by the “one-dimensional Newton method”.

4. APPLICATIONS AND RESULTS

We have developed a prototype tool using the Java programming language and Java3D class library.

The following screenshots show the capacity of our algorithm to identify the main interest of a complex system by highlighting the global view and to draw the hierarchical structure.

Our algorithm has the same scalability problem as the other force-directed layout techniques. An improvement will be added in the future to make the algorithm more scalable.

In practice, a complex system is rarely structured as a complete graph. It is not a pure tree either. Therefore it is sometimes possible to create a partition of the set of vertices according to their weights. Here, we mean by weight the number of edges incident from a vertex. We simply use the partition of the graph when it is possible and necessary to show the structure and to improve the layout, but it is not the main interest of our method.

By grouping the vertices according to their weight, the graph can be subdivided into several hierarchies as follows: H_1:{O_1, O_2, O_3,…,O_n}, H_2: {{O_{11},O_{12},…},{O_{21},O_{22},…},…,{O_{h1}, O_{h2},…}}, H_3, H_4, …

Once vertices are hierarchically grouped, the first hierarchy, here H_1, is then given an initial layout. Its elements are distributed in the initial sphere (S) and their positions are refined using the force-directed method. After this step, each vertex O_i of H_1 is placed at the centre of a sphere S_i. The next step starts by distributing in the sphere S_i the vertices (O_{i1}, O_{i2}, ...) considered as the sub-hierarchy of O_i and the force-directed method is then applied. This process is recursively applied through all the hierarchies until positioning all vertices.

In the case where it is impossible to subdivide the graph, vertices are directly distributed in the largest sphere inscribed in the embedding area.

This process applies easily to trees because we just need to place the root vertex at the centre of the initial sphere and to recursively subdivide this sphere through the remaining hierarchies until the leaf vertices.

The six following figures represent the general structures of large systems, i.e. the combination of a tree and a general graph. Figure 1 shows how fifteen vertices are placed equitably in the area and figure 2 represents them with maximum links, i.e. a vertex is linked to every other ones. Figure 3, 4 and 5 show the typical case of a mixture of trees and general graphs. Figure 6 specifically represents a pure tree rooted from the vertex placed at the centre of the area. The embedding area is represented by the xyz axes and colored in grey. We have assigned different colors to vertices in order to highlight hierarchical levels in the graph.

We have developed a simple user interface as a test bed for our algorithm. The number of nodes on each level of a graph can be specified to test the rapidity of the drawing. Thus it is easy to build all types of graph and analyze the strength of the algorithm. The three standard ways of navigation are also added to allow the study of a graph. All the details of a complex graph can then be shown by zooming on in, rotating or translating the display.

Our algorithm does not take into account the effects of edges; they are just used to calculate the weight of each vertex and only plotted after positioning all vertices.

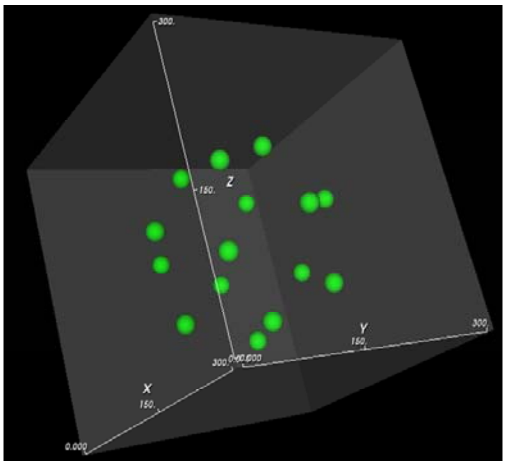

Figure 1: 15 vertices

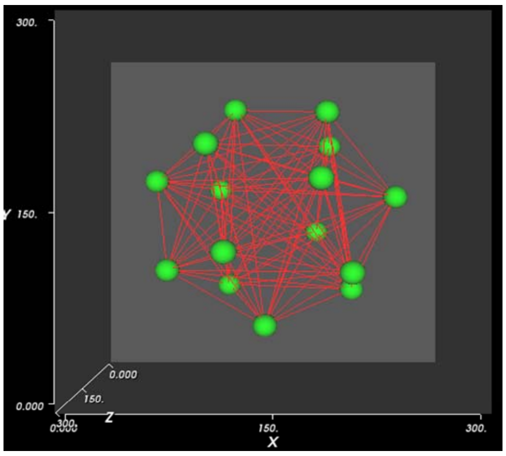

Figure 2: 15 linked vertices

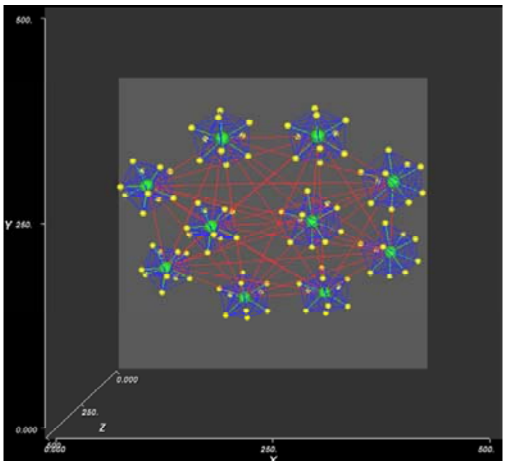

Figure 3: 110 vertices

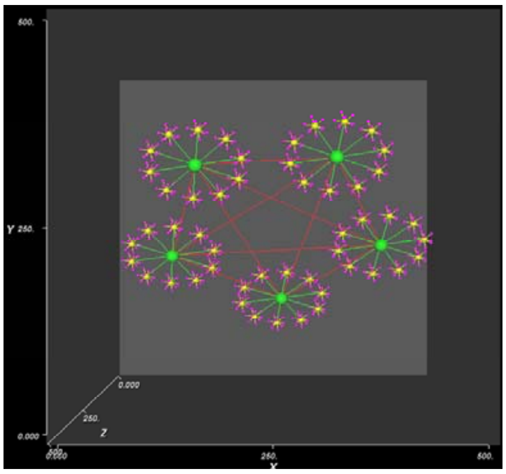

Figure 4: 550 vertices

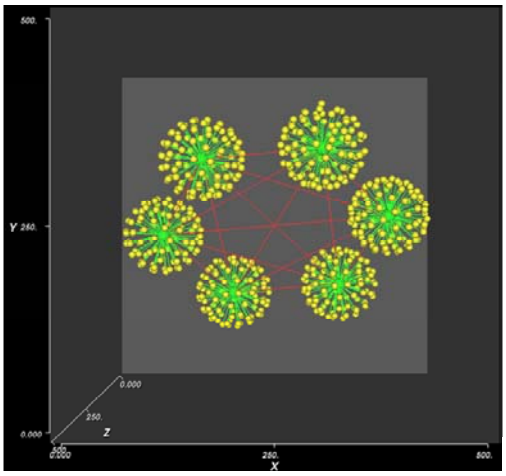

Figure 5: 606 vertices

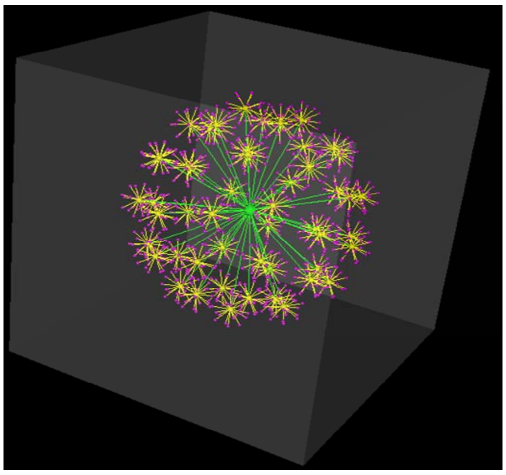

Figure 6: 1051 vertices

Figures 3, 4 and 5 show the typical case of a complete graph. They respectively contain 110, 550 and 606 nodes and the drawing took respectively 0.30, 1.30 and 2.0 seconds on a laptop with a 1Gh AMD processor.

Generating the layout of a tree is very fast because the initial sphere containing the route node is successively subdivided into different branches. Figure 6 for example represents 1051 nodes and is computed in around 2.0 seconds.

Compared with different force-directed algorithms our algorithm can draw a large graph in a reasonable time because it is based on the natural mechanism that consists in associating with a hierarchical of graphs and the drawing starts by the smallest graph in the hierarchy and drawing larger and larger graphs. The second advantage of our method is the use of a numerical optimization while computing the displacement of a vertex.

5. CONCLUSIONS

We have presented a new method to optimize the force-directed placement algorithm. The conjugated gradient of Fletcher-Reeves is implemented to perform the numerical optimization. Our method may not be the fastest amongst the force-directed placement algorithms but its main advantage is its effectiveness in minimizing total energy while most algorithms based on this method fail and do not clearly express their cost function. Our algorithm is addressed to three dimensional drawings of general undirected graphs and trees.

To avoid collisions between vertices we consider the largest sphere inscribed in space and subdivide it into smaller spheres, each containing a vertex. The unused space is minimized by improving the position of each vertex with the force-directed method.

According to the structure of a graph it is possible to associate a partition method in order to improve the layout.

Our next goal consists in improving the presentation by introducing the notion of weighting when assigning space to vertices. Indeed, some vertices have more descendents than others. The size of the area assigned to each vertex must then be proportional to the number of its descendents. Therefore, the presentation will be much more interesting when introducing the vertices' weight into the energy function.

Our algorithm can be used in many computer science fields such as large-scale computing network, telecommunication networks' display, but we are going to apply it to our project of large computing networks 3D visualization.

6. REFERENCES

[Bar86] J.Barnes and T.Hut, A Hierarchical O(N log N) force-calculation algorithm, Nature, vol.324, pp.446-449, 1986.

[Bat98a] G. Di Battista, P. Eades, R.Tamassia and I. G. Tollis, 1998. Graph Drawing: Algorithms for the Visualization of Graphs, Prentice-Hall, Englewood Cliffs.

[Chu01a] N. Churcher, Alan Creek, 2001. Building Virtual Worlds with the Bing-Bang Model. *Australian Symposium on Information Visualization*, Sydney.

[Dan98a] Daniel Tunkelang, 1998. JIGGLE: Java Interactive Graph Layout Environment. In Whitesides, Sue H., Eds. In *Proceedings Graph Drawing*, pages pp. 413-422, Montréal, Canada

[Dav96a] R. Davidson, D. Harel, 1996. Graph Drawing Nicely using simulated Annealing. *ACM Trans. Graphics*, pp. 301-331.

[Dwy01a] T. Dwyer, 2001. Three Dimensional UML Using Force Directed Layout. *Australian Symposium on Information Visualization*, Sydney.

[Ead84a] P. Eades, 1984. A Heuristic for Graph Drawing. *Congressus Numerantium*, pp. 149-160.

[Fle64a] R. Fletcher and C.M Reeves, 1964. Function minimization by conjugate gradients. In *Computer Journal 7,* pp. 149-154.N.

[Fru91a] T. Fruchterman, E. Reingold, 1991. Graph drawing by force-directed placement. *Software. – Practice and Experience*, pp. 1129–1164.

[Gaj04a] P. Gajer, M. T. Goodrich, S. G. Kobourov, 2004. A multi-dimensional approach to force-directed layouts of large graphs. *Computational Geometry, Theory and Applications*, pp. 3-18.

[Gil90a] J. C. Gilbert, Nocedal J, 1990. Global convergence properties of Conjugated Gradient methods for optimization. *Research Report*. National Institute of Computer Science and Automatic (INRIA).

[Har02a] D. Harel, Y. Koren, 2002. A Fast Multi-Scale Algorithm for Drawing Large Graphs. *Journal of Graph Algorithms and Applications*, pp. 179-202.

[Kam89a] T. Kamada, S. Kawai, 1989. An algorithm for drawing general undirected graphs. In *Information Processing Letters*, 31, pp. 7–15.

[Ngu02a] Q. V. Nguyen, M. L. Huang, 2002. Using space- optimized tree visualization for web site-mapping. *International Conference on Internet Computing* (IC'02), Las Vegas, Nevada, USA, pp. 622-628.

[Wal03a] C. Walshaw, 2003. A Multilevel Algorithm for Force-Directed Graph Drawing. *Journal of Graph Algorithms and Applications*, http://jgaa.info/vol. 7 no. 3, pp. 253-285.

[Wan05a] C.Y Wang, M.X. Li, 2005. Convergence property of the Fletcher-Reeves Conjugated Gradient method with errors. In *Journal of Industrial and management optimization*, Volume 1, Number 2.