International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:1, No:3, 2007

A Force-directed Graph Drawing based on the Hierarchical Individual Timestep Method

T. Matsubayashi, and T. Yamada

Abstract**—** In this paper, we propose a fast and efficient method for drawing very large-scale graph data. The conventional force-directed method proposed by Fruchterman and Rheingold (FR method) is well-known. It defines repulsive forces between every pair of nodes and attractive forces between connected nodes on a edge and calculates corresponding potential energy. An optimal layout is obtained by iteratively updating node positions to minimize the potential energy. Here, the positions of the nodes are updated every global timestep at the same time. In the proposed method, each node has its own individual time and timestep, and nodes are updated at different frequencies depending on the local situation. The proposed method is inspired by the hierarchical individual timestep method used for the high accuracy calculations for dense particle fields such as star clusters in astrophysical dynamics. Experiments show that the proposed method outperforms the original FR method in both speed and accuracy. We implement the proposed method on the MDGRAPE-3 PCI-X special purpose parallel computer and realize a speed enhancement of several hundred times.

Keywords**—** visualization, graph drawing, Internet Map

I. Introduction

IN many scientific and engineering domains, complicated relational data structures are frequently represented by networks or, equivalently, graphs. For example, WWW (World Wide Web) sites are often represented by *hyperlink networks*, with pages as nodes and hyperlinks between pages as edges, the interactions between genes, proteins, metabolites and other small molecules in an organism are represented by *gene regulatory networks*, and the relationships between people and other social entities are characterized by *social networks*. This popularity is because network representations often provide important insights to researchers in understanding the intrinsic data structure with the help of some mathematical tools such as graph theory, as well as by examining an embedded layout in a low-dimensional Euclidean space. Embedding a graph in a low-dimensional Euclidean space, which is called graph-drawing, is especially useful when the graph is sparse as most of the real world graph data are.

In this paper, we propose a fast and efficient method for drawing undirected large-scale graph data based on the force-directed method proposed by Fruchterman and Rheingold (1991)[3]. In their method, an optimal layout is obtained by iteratively updating node positions to minimize the potential energy. Here, the all positions of the nodes are updated in every step at the same time. Our proposal, on the other hand, gives each node its own individual time and timestep, and nodes are updated with a different frequency depending on the local situation. The proposed method is inspired by the hierarchical individual timestep method used in astrophysical dynamics. We have implemented the proposed method on the MDGRAPE-3 PCI-X special purpose parallel computer, and succeeded in achieving a speedup of several hundred times. Thus, the proposed method with parallel processing enables us to visualize graphs that have more than 10^4 nodes N and edges E.

The organization of the paper is as follows. In §II, we introduce some related works. In §III, we describe the suggested algorithm and its updating method. In §IV, we describe the results of simulations. The summary is presented in §V.

T. Matsubayashi is with the Communication Science Laboratories, NTT Corporation, 2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan (corresponding author to provide phone: +81-774-93-5360; fax: +81-774-93-5155; e-mail:tatsushi@cslab.kecl.ntt.co.jp

T. Yamada is also with the Communication Science Laboratories, NTT Corporation (e-mail: yamada@cslab.kecl.ntt.co.jp)

II. Related work

Various methods have been proposed to solve the graph layout problem since the 1980's. Among them, the most successful strategy is the "force-directed method". The basic idea of the force-directed method is to simulate a system of natural forces and find the minimum energy state of the system by updating coordinates (node positions) to reflect the direction of the force computed based on the graph distance, geodesic distance between nodes on the graph, or some other adjacency measure. This corresponds to finding the zero temperature state of a crystal structure in molecular dynamics. In this section, we briefly review related research in the force-directed method literature.

A. Spring Embedder method

The most popular method for drawing undirected graphs is the so-called *spring method*. The first spring method, called the spring embedder method, was proposed by Eades (1984) [1]. This method likens a graph to a mechanical collection of rings (the nodes) and connecting springs (the edges). Two connected rings are attracted to each other or repelled from each other according to their distance and the properties of the connecting spring. A state with minimum energy in the springs corresponds to a nice drawing of the underlaying graph.

B. Kamada & Kawai Method

Kamada and Kawai (1989) [2] proposed another spring method (hereafter referred to as the KK method). In the KK method, a graph is modeled as a system of springs spanning nodes; their natural lengths are proportional to the graph

distance of the nodes. Let $\mathbf{x}_i$ be the position of node i in the given graph layout. The spring force on node i is defined as follows:

$$\mathbf{a}_{KK,i} = \sum_{j \neq i}^{N} k_{ij} \left(1 - \frac{l_{ij}}{|\mathbf{x}_{ij}|}\right) \mathbf{x}_{ij} \ , \tag{1}$$

where $|\mathbf{x}_{ij}| = |\mathbf{x}_i - \mathbf{x}_j|$ and l_{ij} are the Euclidean distance and the graph distance between node i and j, respectively. The spring constant, k_{ij}, is defined as $k_{ij} = k/l_{ij}^2$, where k is the normalization factor.

The KK method sequentially updates the node positions based on the Newton-Raphson method: a node in the highest energy state is selected and its positions, and then the total energy, are updated. On the other hand, the FR method that will be described later updates all node positions at once. The KK method requires $O(N^2)$ memory space to store l_{ij} matrix. Even modern general-purpose computers, which can have memory resources of several Giga bytes, may suffer system memory overflow when $N \geq 10^5$. Thus, the KK method is not applicable to large-scale graph data.

C. Fruchterman & Reingold Method

Fruchterman and Reingold proposed a method (hereafter referred to as the FR method) based on the attractive force $\mathbf{a}_a$ between the nodes on an edge (i.e., between connected nodes) as well as the repulsive force $\mathbf{a}_r$ between all (connected or unconnected) nodes defined as follows:

$$\mathbf{a}_{a,i} = \frac{1}{k} \sum_{j \neq i}^{E_i} |\mathbf{x}_{ij}| \mathbf{x}_{ij} \ , \tag{2}$$

$$\mathbf{a}_{r,i} = -k^2 \sum_{j \neq i}^{N} \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|^2} \ , \tag{3}$$

where E_i is the number of edges connected from node i, and k is the normalization factor. The sum of force $\mathbf{a}_i$ for node i is derived from equations(2) and (3) as $\mathbf{a}_i = \mathbf{a}_{a,i} + \mathbf{a}_{r,i}$.

As mentioned before, the FR method updates the positions of all the nodes simultaneously with equal frequencies by moving them toward the direction of the force given by equation (2). Here, the maximum step size is upper bounded by the cooling function $t_{\rm cool}$ as follows:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{a}_i \times \min\left(1, t_{\rm cool}(t)/|\mathbf{a}_i|\right). \tag{4}$$

The cooling function $t_{\rm cool}(t)$ is defined as

$$t_{\rm cool}(t) = C \times (1 - t/T_{\rm END}) \ . \tag{5}$$

where C is the initial maximum step size. In this method, we have to determine the appropriate values for C and $T_{\rm END}$ for a given data set before starting a simulation. We will discuss this issue in more detail in §IV-A. An outline of node update is shown below.

(a) Set initial positions randomly.
(b) Calculate force $\mathbf{a}_i$ on each node i.
(c) Update $t_{\rm cool}$.
(d) Calculate $\mathbf{x}_i(t+1)$ on each node i.
(e) Update the time, $t = t + 1$.
(f) Go back to step (b), until $t = T_{\rm END}$.

D. Recent works

The conventional methods described in the previous section require computation of the interaction between all nodes, thus, the computational order is $O(N^2)$. However, several approximation techniques have been proposed in recent years.

Quigley and Eades(2000) [4] proposed an efficient method using Burns-Hut tree code [Burns and Hut(1986)[5]]. The Barnes-Hut tree code is an approximation algorithm developed for astrophysical dynamics. In this method, the forces from distant nodes are grouped together and computed as a joint force from their barycenter. This reduces the computational cost for each node interaction from $O(N^2)$ to $O(N \log N)$.

Adai et al.(2004) [6] proposed LGL (Large Graph Layout) method for drawing large-scale networks. In their method, the minimum spanning-tree is constructed from the link structure of the original network data, and the coordinates are decided according to the minimum spanning-tree. Recently, the graph-drawing project called "Opte Project" has visualized the largest Internet network mapping [7]. It examined 5 million nodes and 50 million edges, and took 252.68 hours to compute. However, the quality of the results is questionable, because the LGL method disregards a certain number of the network connections that are not the part of the minimum spanning-tree. This problem will be discussed again in § IV-B.

III. New Algorithm

The primary focus of the conventional methods described above is the definition of the forces, and their efficient calculations. Another challenge is to speed up the optimization process.

In this paper, we propose a new method in which each node has an individual time and timestep based on the "individual timestep method". The FR method uses the Shared timestep with which all nodes update simultaneously. The proposed method is based on the FR method. Hereafter, we refer to the proposed method as the FR-HI method. The individual timestep method was originally developed to realize the high-accuracy and high speed N-body calculations needed for astrophysical dynamics [8]-[10], especially for a star cluster or cluster of galaxies where density contrast is large. Figure 1 shows pictorial views of time evolution in the shared and individual timestep schemes, respectively.

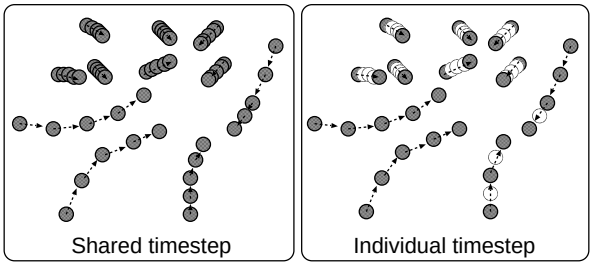

Fig. 1. The left and right figures show a the schematic figure of time evolution for the shared and individual timestep scheme, respectively. For Individual timestep, longer timestep nodes are calculated as predictor ◯.

In the shared timestep method, even if the force on a node is almost zero and its effect on the coordinates of the node is

small, the coordinates and the value of the force are updated every timestep. On the other hand, in the individual timestep method, every node has its own individual time which is different from the system-wide global time. When the force on a node is strong, then the individual time of the node becomes short, and the node is updated frequently. When the force is weak, the individual time becomes long and the node is updated only infrequently, so the computational cost can be reduced. Moreover, the forces from the other nodes are calculated by using the approximated coordinates which we call "predictor".

If node updates are completely independent, the cost of the predictor calculations becomes large. The solution is the hierarchical timestep; the timesteps are quantized so that nodes are hierarchically structured which lowers the cost calculating the predictors. Moreover, it enables the use of parallel processing. We will discuss parallel processing on a special purpose computer in §III-D. The next section examines the Hierarchical Individual timestep method in detail.

A. Hierarchical Individual timestep method

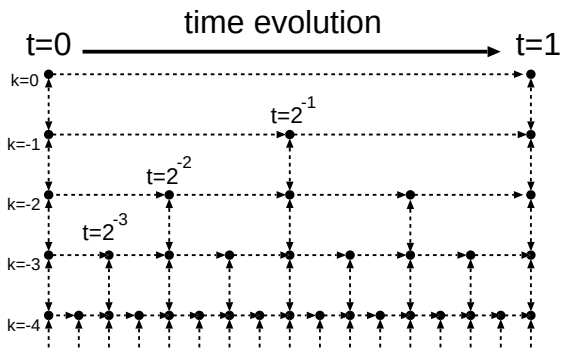

Fig. 2. Schematic view of time evolution for hierarchical individual timestep method. All nodes have individual time on • of this figure, and update when they quantize with global time t. Here, the condition for the change of Δt_i is defined by Eq.(7)-Eq.(10). Thus, node states can move only on the dotted lines.

In the FR-HI method, each node i has its own (individual) time t_i and timestep Δt_i, which is rounded to an integer. Its coordinates are updated when $t = t_i$ (i.e., when t_i is "synchronized" with the global time t), as follows:

$$\mathbf{x}_i(t_i + \Delta t_i) = \mathbf{x}_i(t_i) + \Delta t_i \times \mathbf{a}_i(t_i). \tag{6}$$

We would like to make Δt_i inversely proportional to the force. However, it is not computationally efficient if each node has completely different timestep size. Therefore, the essential idea of the proposed scheme is to adjust the timestep of each node so that it satisfies

$$\Delta t_i = 2^k \quad \text{if } 2^k \leq \frac{\eta}{|\mathbf{a}_i(t_i)|} < 2^{k+1}, \tag{7}$$

where k is an integer, and η is a dimensionless constant that controls the computational accuracy. In the experiments described in this paper, we always use $\eta = 1.0$. By this adjustment, the timestep size of each node is "quantized" so that nodes with similar force strength share the same time and are updated simultaneously. As a result, we update the position of node i at time t, such that

$$t \bmod 2^k = 0 \ , \tag{8}$$

where "mod" is the modulo operator.

When we update the position of node i at time t, we may also need to update the timestep itself. We first calculate $\Delta\tilde{t} = \eta/|\mathbf{a}_i(t_i)|$, the non-quantized version of the new timestep. The new timestep $\Delta\ t_{i,new} = 2^{k_{new}}$ is calculated as

$$k_{\rm new} = \begin{cases} k-1 & \text{if } \Delta\tilde{t}_i < 2^k \\ k+1 & \text{if } \Delta\tilde{t}_i > 2^{k+1} \ \text{ and } \ t \bmod 2^{k+1} = 0 \\ k & \text{otherwise .} \end{cases} \tag{9}$$

Note that even when the non-quantized version of the timestep is increased as $\Delta\tilde{t}_i > 2^{k+1}$, the timestep is not updated from 2^k to $\Delta\ t_{i,new} = 2^{k_{new}} = 2^{k+1}$ if t does not satisfy $t \bmod 2^{k+1} = 0$.

Figure 2 shows schematic view of the time evolution for the hierarchical individual timestep method. All nodes have individual time on "•" in this figure, and are updated when global time t is divisible by Δt_i. Here, Δt_i is updated when the conditions defined as Eq.(7)-Eq.(9) are satisfied. Thus, node states can move along the dotted lines.

In addition, we set upper limit $\Delta t_{\rm max}$ and lower limit $\Delta t_{\rm min}$ to the update timestep size. We force node synchronization by setting the lower limit of the timestep. The role of the lower limit is similar to that of the maximum temperature in the FR method. It prevents divergence of coordinate values using a force given from initial random placement. If this lower limit is not used, the progress of the calculation dramatically slows. Thus, when $\Delta t_i = \Delta t_{\rm min}$, the dynamic formula is replaced by

$$x_i(t_i + \Delta t_{\rm min}) = x_i(t_i) + \frac{\eta}{|\mathbf{a}_i(t_i)|} \times \mathbf{a}_i(t_i) \tag{10}$$

In this paper, we set $\Delta t_{\rm max} = 1$, $\Delta t_{\rm min} = 2^{-10}$.

B. Updating method

In the FR-HI method, only the nodes whose time $t_i + \Delta\ t_i$ is synchronized with global time, t, are updated. Where, the global time is selected as

$$t = \min\ (t_i + \Delta t_i). \tag{11}$$

We set the number of synchronized nodes as

$$n_s(t) = \sum_i^N \ \begin{cases} 1 & \text{if } \ t = t_i + \Delta\ t_i \\ 0 & \text{otherwise .} \end{cases} \tag{12}$$

Thus, the computational cost for each node interaction is reduced from $O(N^2)$ to $O(n_s(t) \times N)$.

The force on a node is calculated based on the predicted position (predictor) of other nodes. The predicted position is updated from t_j to t as

$$\tilde{\mathbf{x}}_j(t) = \mathbf{x}_j(t_j) + (t - t_j) \times \mathbf{a}_j(t_j). \tag{13}$$

where j runs through all nodes. The predictor of our scheme requires no memory of the previous timesteps.

Using the force yielded by equations (2) and (3), we evaluate the force at t for the synchronized nodes as follows:

$$\mathbf{a}_i(t) = \frac{1}{k}\sum_{j\neq i}^{E_i} |\mathbf{x}_{ij}|\mathbf{x}_{ij} - k^2 \sum_{j\neq i}^{N} \frac{\mathbf{x}_{ij}}{(\mathbf{x}_{ij}^2 + \epsilon^2)} \ , \tag{14}$$

where

$$\mathbf{x}_{ij} = \tilde{\mathbf{x}}_j - \tilde{\mathbf{x}}_i, \tag{15}$$

and we apply the softening parameter ϵ to teat repulsive forces between nodes so that the calculation does not diverge. In this paper, we set $\epsilon = 0.01$.

The updating process proceeds according to the following steps:

(a) Set initial position randomly, with $t_i = 0$ for all nodes.
(b) Calculate force $\mathbf{a}_i$ for all nodes.
(c) Calculate Δt_i for all nodes.
(d) Set the global time t to be minimum $t_i + \Delta t_i$.
(e) Select the synchronized nodes with $t = t_i + \Delta t_i$, and set $n_s(t)$.
(f) Predict position $\tilde{\mathbf{x}}_i$ of all nodes at time t.
(g) Calculate force $\mathbf{a}_i$ for the synchronized nodes.
(h) Update position $\mathbf{x}_i$ for the synchronized nodes.
(i) Update time $t_i = t_i + \Delta t_i$ of the synchronized nodes.
(j) Calculate timestep Δt_i of the synchronized nodes.
(k) Go back to step (d), until $t \geq T_{\rm END}$.

C. Evaluation method

1) System Energy Ψ: In this paper, we use the scalar potential energy of the system to evaluate the accuracy of the algorithms. The Force-directed methods are equivalent to determining the crystallized state of network data. Since the nodes will be always updated in the accelerated directions, the ideal final states correspond to the lowest potential energy states. Thus we define the scalar potential energy function to satisfy the following gradient relationship equation

$$\mathbf{a} = \nabla\Psi \ . \tag{16}$$

Using equation (14), the total energy Ψ is given by

$$\Psi = \frac{1}{3k}\sum_{i,j}^{E} |\mathbf{x}_{ij}|^3 - \frac{k^2}{2}\sum_{i}^{N}\sum_{i\neq j}^{N} \log\left|\mathbf{x}_{ij}^2 + \epsilon^2\right|^{1/2} \ . \tag{17}$$

Hereafter, we use energy Ψ as the objective function.

2) Averaged Steps n_{ave}: Next, to compare the FR-HI method to the FR method, we use the averaged number of steps, n_{ave}, defined as

$$n_{ave} = \frac{1}{N}\sum_{t=0}^{T} n_s(t) \ . \tag{18}$$

Actually, it is difficult to accurately evaluate the computational complexity to compare our method to the FR method. However, it is useful to use n_{ave} because over 99.9% of the time is spent calculating the part of $O(N^2)$.

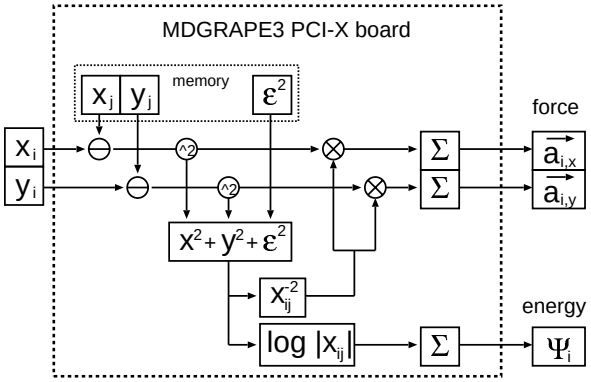

Fig. 3. The pipeline image of the architecture of MDGRAPE-3.

D. Special Purpose Computer: MDGRAPE-3

The primary advantage of parallel algorithms like the FR-HI method (also FR method) is that they allow vector pipelined machines to be used more efficiently. The total cost calculating the repulsive force, equations (2), (3), (14), and (17), is $O(N^2)$. Therefore, in a large scale network, this calculation takes more than 99% of the total time. Thus we use MDGRAPE-3 PCI-X [1]to determine the repulsive force and the energy calculation, and the other calculations are done on a host computer. Figure 3 shows the pipeline image of the architecture of the MDGRAPE-3 chip.

As an example, to calculate model D (see table I) by the FR-HI method, the user CPU time is about 70000 minutes if we use only the host machine (Intel Xeon 3.2GHz). On the other hand, it takes only 348 minutes by using MDGRAPE-3. Actually, over 99.9% of the time is spent calculating the acceleration and energy in the host machine. We implemented the FR-HI method on MDGRAPE-3, and realized a speedup of several hundred times. Thus, to calculate the interactions, the use of parallel processing and a special purpose computer is very effective and efficient.

IV. Results

We examined the IP network data obtained from the Opte Project, which was discussed in §II-D, in experiments. The nodes in these data are Class-C network addresses, and the edges represent the connections based on the routing information of the network obtained by the *traceroute* command. The four network data sets shown in Table I were used in this paper.

A. Effect of $T_{\rm END}$

One of differences of the FR-HI method from the FR method is that we do not need to set $T_{\rm END}$ explicitly. In the

[1]MDGRAPE-3 won the Honorable Mention of the IEEE Gordon Bell Prize (2006)[11]. MDGRAPE-3 (Molecular Dynamics GRAPE) is a specialized computer that was designed for the scalar and vector calculations involving the summation of a function of particle distances. It can also handle three dimensions. The MDGRAPE-3 PCI-X processor board is a general purpose version; it operates at 250 MHz and its performance reaches 330 Gflops with 2chips.

TABLE I
INTERNET DATA

model	number of nodes (N)	number of edges (E)
A	35638	42827
B	35836	42387
C	40027	47215
D	134023	161283

FR method, it is necessary to set $T_{\rm END}$ to control the energy to ensure convergence into the lower state. However, it is difficult to know an appropriate $T_{\rm END}$ beforehand, especially when dealing with a large-scale network. When energy Ψ doesn't converge well enough, it is necessary to restart the long calculation.

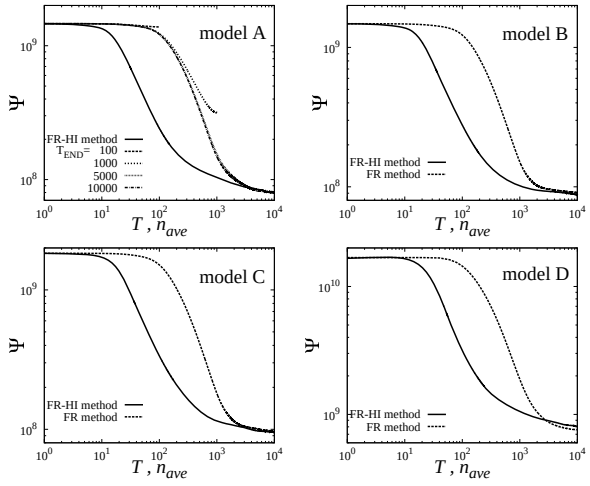

Fig. 4. Simulation results for models A-C with FR and FR-HI methods. The left top shows the parameter effect on the results of the FR method, $T_{\rm END} = 100, 1000, 5000, 10000$.

The left top of Figure 4 shows the parameter effect on the results in the FR method. The lines plot the FR-HI and the FR method responses with $T_{\rm END} = 100, 1000$, and 10000. From the figure, it is clear that the energy converges to a low enough value only when $T_{\rm END} \geq 10000$. The results of the FR-HI method, shown by the solid lines, indicate that the FR-HI method offers faster convergence than the original FR method. The calculation results for models B-D are also shown in Figure 4; the results are similar to those of model A, where we set $T_{\rm END} = 10000$ for all models in the FR method.

Figures 5 and 6 show the graph layout results for model A obtained by the FR method, and the FR-HI method respectively. The overall structural characteristics are not clearly seen with the FR method until $T_{\rm END} \simeq 10000$, while it can be seen from a very early stage with the FR-HI method. This is because the hierarchical time step calculation assigns higher updating frequency to more dense subnetworks in the FR-HI method. Thus, we can see that the overall structural characteristics of the network emerge faster with the FR-HI method than with the FR method.

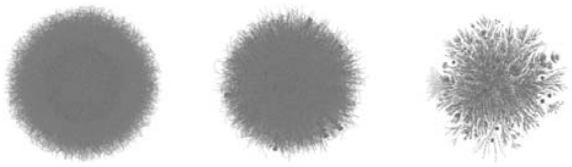

Fig. 5. Graph layout results of FR method. From left to right $T_{\rm END} = 100, 1000, 10000$.

Fig. 6. Graph layout results of proposed method. From left to right $T_{\rm END} = 100, 1000, 10000$.

B. LGL method

As explained in the former paragraph, the force-directed formulation used in the FR and the FR-HI methods is suitable for parallelization. By implementing the proposed method on a parallel computer, it is now possible, for the first time to the best of our knowledge, to obtain a highly accurate graph-layout of very large-scale networks such as the ones given in Table I. We have implemented the proposed method on MDGAPE-3, a special purpose parallel computer. In the LGL method used in OpteProject, a large number of edges of the network are removed in the initial operation to construct a minimum spanning-tree for efficient calculation. However, this results in a poor quality graph layout in that some related nodes that should be put closely together tend to be widely separated.

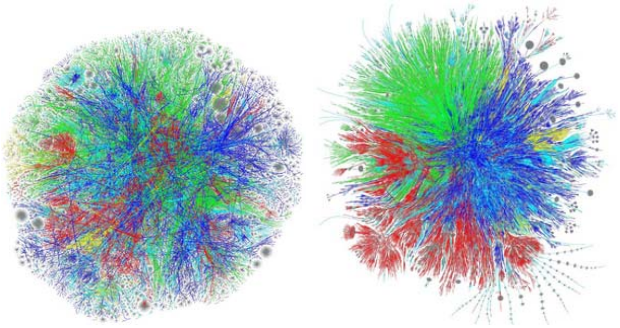

Fig. 7. Left and Right are the layout results of model D with the LGL method and FR-HI method, respectively. The color classification of the edges are according to the domain region as "Asia Pacific - Red", "Europe/Middle East/Central Asia/Africa - Green", "North America - Blue", "Latin American and Caribbean - Yellow", "RFC1918 IP Addresses - Cyan", and "Unknown - gray".

Figure7 shows the layout results of model D obtained by the LGL method and FR-HI method. The total user CPU time taken for the calculation was 400 minutes and 300 minutes for the LGL method and the FR-HI method, respectively. Different colors were assigned to the edges according to the domain

region of the nodes. It can be seen from this figure that, with the LGL method, although the edges of the same color that are from the same region should be placed close together, they are spread all over, which may be due to the initial truncation operation. On the other hand, it is clear that the FR-HI method places these classified edges in the same neighborhood.

TABLE II
AVERAGED LENGTH AND DISPERSION OF THE NODES

method	L_{box}	$\langle \mathbf{x}_{ij} \rangle / L_{box}$	$\sigma / \langle \mathbf{x}_{ij} \rangle$
LGL	188.64	0.0103	1.551
FR-HI	340.48	0.0120	0.575

Next, we evaluated the quality of the graph layout results by the histogram of the edge length. The second column of Table II shows the maximum width of the coordinate L_{box} (corresponds to the vertical or horizontal size of Figure7). The third column of Table II shows the averaged length of all edges normalized by L_{box}. The fourth column of the table, σ, indicates the dispersion of $|\mathbf{x}_{ij}|$. From this table, it is clear that the LGL method has larger normalized dispersion than the FR-HI method.

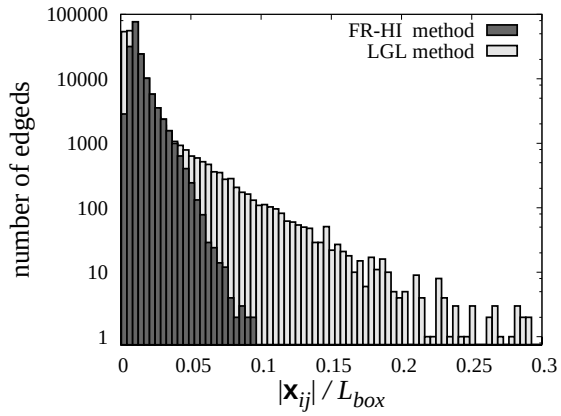

Fig. 8. Histogram of the edge lengths of the LGL method and FR-HI method. Horizontal axis indicates edge length $|\mathbf{x}_{ij}|$ in which L_{box} is delimited by 250 classes, and the length of each edge is normalized by L_{box}. The vertical axis indicates the number of edges in each class.

Figure 8 shows the histograms of the normalized edge length $|\mathbf{x}_{ij}|/L_{box}$ for the LGL method and FR-HI method. It can be seen that the LGL method yields more long edges than the FR-HI method, which indicates that the layout yielded by the LGL method does not respect the adjacent nodes present in the original data. With the FR-HI method, the longest edge is less than 10% of L_{box}. On the other hand, with the LGL method, the longest edge is about 30% of L_{box}, and the number of edges that are over 10% length of L_{box} is about 1% of the total number of edges.

Therefore, methods that consider the effect of all node relations like the FR method or FR-HI method place connected nodes together while the LGL method separates them. This problem indicates that scheme based on the minimum spanning-tree have limited ability in creating accurate graph layouts.

V. SUMMARY

The FR-HI method and the FR method have two main differences. The first is the node updating process. In the FR-HI method, each node has an individual time and timestep and updating is hierarchical, while the updating process in the FR method is always simultaneously. Therefore, the FR-HI method offers much higher speeds and a more efficient updating process. The second is the setting of T_{END}. In the FR-method, the graph layout result depends on T_{END} as shown in Figure 5. However, in the FR-HI method, there is no necessity for setting T_{END} beforehand and then re-performing the calculations.

The LGL method is widely used to create large-scale graph layouts, and it is an efficient and fast method. However, its weakness is that connected nodes can become separated. Therefore, we conclude that a faster and more efficient method like the FR-HI method on a parallel processing machine like MDGRAPE-3 is necessary.

In this research, to evaluate the efficiency of selecting the update nodes, we calculate the acceleration from all nodes. However, the FR-HI method can support the approximation method like Tree-code which is used in the FADE method. to make the calculations even faster. As future work, we will examine the combination of the FR-HI method with other approximation techniques.

REFERENCES

[1] Eades, P.: A heuristic for graph drawing, Congresses Numerantium, 42, 149-160 (1984)
[2] Kamada, T., and Kawai, S.: An algorithm for drawing general undirected graphs, Information Processing Letters, 12, 31, 7-15 (1989)
[3] Fruchterman, T. M. J., and Reingold, E. M.: Graph Drawing by Force-directed Placement, Software - Practice and Experience, 11, 21, 1129-1164 (1991)
[4] Quigley, A., and Eades, P.: FADE:Graph Drawing, Clustering, and Visual Abstraction, Proceedings of Graph Drawing 2000, Lecture Notes in computer Science, 1984, 183-196 (2001)
[5] Barnes, J., and Hut, P.: A hierarchical $O(N \log N)$ force-calculation algorithm, *Nature*, 04, 324, 446-449 (1986)
[6] Adai, A. T., Date, S. V., Wieland, S., and Marcotte, E. M.: LGL: Creating a map of protein function with an algorithm for visualizing very large biological networks. *Journal of Molecular Biology*, 340(1):179?190, June 2004.
[7] Lyon, B.: The Opte Project(2005) http://www.opte.org/
[8] Ahmad, A., and Cohen, L.: A numerical integration scheme for the N-body gravitational problem, Journal of Computational Physics, 12, 389 (1973)
[9] McMillan, S. L. W.: The Vectorization of Small-N Integrators, *Lecture Notes in Physics*, 267, 156 (1986)
[10] Makino, J.: A Modified Aarseth Code for GRAPE and Vector Processors, *Publications of the Astronomical Society of Japan*, 43, 859-876 (1991)
[11] Narumi, T., Ohno, Y., Okimoto, N., Koishi, T., Suenaga, A., Futatsugi, N., Yanai, R., Himeno, R., Fujikawa, S., Ikei, M., and Taiji, M.,: A 55 TFLOPS Simulation of Amyloid-forming Peptides from Yeast Prion Sup35 with the Specialpurpose Computer System MDGRAPE-3, Proceedings of the SC06 (High Performance Computing, Networking, Storage and Analysis), CDROM, Tampa, USA, Nov. (2006)