Infinite Trees and the Future [*]

Camil Demetrescu[†] Giuseppe Di Battista[‡] Irene Finocchi[§]
Giuseppe Liotta[¶] Maurizio Patrignani[‡] and Maurizio Pizzonia[‡]

Abstract

We study the problem of designing layout facilities for the navigation of an "infinite" graph, i.e. a graph that is so large that its visualization is unfeasible, even by gluing together all the screen snapshots that a user can take during the navigation. We propose a framework for designing layout facilities that support the navigation of an infinite tree. The framework allows to exploit the knowledge of future moves of the user in order to reduce the changes in her mental map during the navigation. Variants of the classical Reingold-Tilford algorithm are presented and their performance is studied both experimentally and analytically.

1 Introduction

Designing layout facilities for the navigation of "infinite" graphs is one of the challenges of the Graph Drawing field. By "infinite" we mean that it would be unfeasible to visualize the graph entirely, even by gluing together all the screen snapshots that a user can take during the navigation. Examples of graphs that can be considered infinite in the above sense come from disparate application areas including World Wide Web, Knowledge Bases, Communication Networks, and Semantic Networks.

Several papers on the visualization of very large graphs appear in the literature. A limited list includes [9, 13, 11, 20, 16, 17, 6]. However, the classical assumption is that the input graph, even if huge, is completely known in advance.

We adopt a different point of view making the assumption, similar to the one in [7], that either the graph is too large to be entirely known or it is practically unfeasible to draw it all. Hence, during the navigation, the user is allowed to see only the content of a limited size window that she can move for exploring the graph. It is clear that such a window cannot be purely geometric: If the graph is so large to be impossible to visualize or even to know, then the idea of moving a window on a pre-computed drawing does not make sense. Therefore, our idea of window is that of a topological window, defined in terms of the structure of the graph. For example, at each time of the navigation the topological window may display the subgraph induced by the vertices that have a constant topological distance from a given vertex, considered as the center of the window.

Hence, the graph is displayed as a sequence of drawings determined by the navigation path followed by the user. Since consecutive drawings in the sequence can have a large overlap, it is essential to devise visualization strategies that preserve the mental map [8, 14, 12, 2] of the user. Algorithms devised to preserve the mental map are usually evaluated in terms of both static and dynamic quality measures [15, 3]. Static measures evaluate standard Graph Drawing aesthetics like number of crossings or area used by the drawing. Dynamic measures evaluate how much a drawing is similar to the one(s) preceding it in the sequence.

Our work starts from two main observations.

[*]Research supported in part by the CNR Project "Geometria Computazionale Robusta con Applicazioni alla Grafica ed al CAD", and by the ESPRIT LTR Project 20244 (ALCOM-IT)

[†]Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, 00198 Roma, Italy. `demetres@dis.uniroma1.it`

[‡]Dipartimento di Informatica e Automazione, Università di Roma Tre, via della Vasca Navale 79, 00146 Roma, Italy. `{gdb,patrigna,pizzonia}@dia.uniroma3.it`

[§]Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", via Salaria 113, 00198 Roma, Italy. `finocchi@dsi.uniroma1.it`

[¶]Dipartimento di Ingegneria Elettronica e dell'Informazione, Università di Perugia, via G. Duranti 93, 06125 Perugia, Italy. `liotta@diei.unipg.it`

- While the display area is bounded by the size of the screen, the size of the portion of graph that the system can know at each instant of time is only bounded by the resources of the computer.
- Very often, even if the graph is infinite, the visit of vertices follows an almost predictable pattern. For example, it can be experimentally verified that most users access a Web site following a limited set of favorite patterns.

This naturally gives rise to the following question. Is it possible to devise a drawing strategy that takes advantage of the knowledge of part of the future navigation moves of the user in order to achieve better performance in terms of dynamic quality measures? In other words, suppose that the user is observing the drawing of a certain subgraph G_1 and suppose to know in advance that the next navigation step will require the visualization of a subgraph G_2; given this knowledge, is it possible to draw G_1 so to reduce the changes in the overlapping portions of the drawings of G_1 and G_2? The idea is to exploit the knowledge of future moves in the current visualization. It is interesting to note that the importance of knowing in advance the sequence of graphs to display within an off-line visualization framework has been discussed in [18].

The main components of our framework are:

- A *visualization window* defining, at each step of the navigation, what is the graph (a portion of the infinite graph) to visualize.
- A *knowledge window* defining, at each step of the navigation, a supergraph of the visualized one. The size of the knowledge window affects the drawing of the visualization window.
- A *favorite neighbor* for each vertex of the infinite graph. The favorite neighbor defines what is the next navigation step that the user is going to perform.

As an application of the above framework, we study the problem of navigating in an infinite tree. In spite of the structural simplicity of trees, designing drawing strategies for navigating in infinite trees offers several experimental and theoretical challenges. Also, the Web is often visualized as a tree-like structure (see e.g. [1, 10]). We devise variants of the classical Reingold-Tilford algorithm [19]. This choice is motivated by the fact that the Reingold-Tilford algorithm is among the most robust, simple, effective, and well known algorithms of Graph Drawing. The main results of this paper can be listed as follows.

- We formally define a framework for designing Graph Drawing facilities that support the navigation of an infinite graph (Section 2). The framework is especially targeted to take into account the knowledge of the future moves of the user.
- We design a set of simple strategies based on the Reingold-Tilford algorithm for visualizing infinite trees (Section 2).
- We perform an experimental analysis of the above strategies (Section 3). The analysis shows that exploiting the knowledge of the future moves of the user is not a trivial task, and that some simple strategies can have more drawbacks than benefits from the knowledge of the future.
- Motivated by the experimental results, we perform a theoretical analysis of the drawing algorithms (Section 4). The analysis both explains our experimental results and allows us to compare the performance of the drawing algorithms as the size of the knowledge window goes to infinity.

Through the paper we make use of standard Graph Drawing terminology [5]. For reasons of space, some proofs are only sketched or omitted in this extended abstract.

2 The Framework and the Drawing Strategies

All our trees are finite subtrees of an infinite tree T and each edge is oriented from the parent to the child. The time of the framework is discrete. Hence, it assumes integer values starting from 0. At instant t the user can look at a *visualization window* of a given positive integer *size* h. The visualization window is a subtree $T_h(t)$ of T rooted at a certain vertex $r(t)$ and induced by the vertices of T at (oriented) distance at most h from $r(t)$. Vertex $r(t)$ is the *point of view* of the user at instant t. If the point of view at instant t is $r(t)$, then the user can move it at instant $t + 1$ to any child of $r(t)$.

A *navigation* of T in any interval $[t_1, t_2]$ of time consists of the path (*visualization path*) $r(t_1), \dots, r(t_2)$. The overlap between two visualization windows at consecutive instants t and

$t+1$ is $T_h(t) \cap T_h(t+1) = T_{h-1}(t+1)$. An overlap between consecutive visualization windows is illustrated in Figure 1.

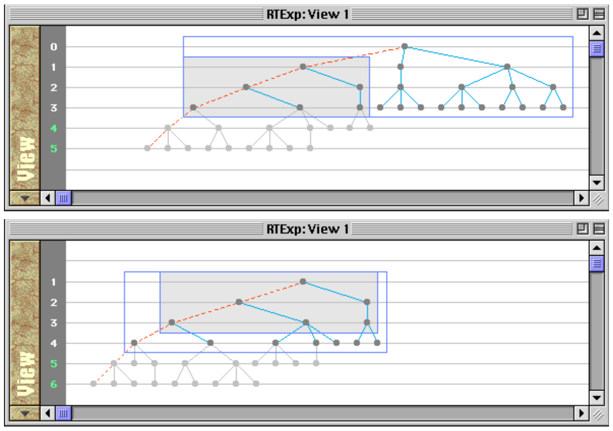


Figure 1: Two successive snapshots produced by the system Leonardo [4] showing the effect of a single navigation step in a randomly generated tree: the light boxes show $T_h(t)$ and $T_h(t+1)$, the shaded boxes show the portion of the tree visible both before and after the navigation step, that is $T_{h-1}(t+1)$. The value of h is 3.

From the point of view of the readability of the visualization, we consider static and dynamic quality measures. The static quality measures evaluate the drawings of $T_h(t)$ and of $T_h(t+1)$ separately. Among the possible static quality measures we focus on the width of the drawing, since it is one of the few degrees of freedom when drawing a rooted tree. The dynamic quality measures evaluate the variations in the overlap of the drawings of $T_h(t)$ and $T_h(t+1)$. Namely, they measure the differences between the drawing of $T_{h-1}(t+1)$ inside the drawing of $T_h(t)$ and the drawing of $T_{h-1}(t+1)$ inside the drawing of $T_h(t+1)$.

In order to optimize the drawing with respect to the dynamic quality measure we exploit the following two observations:

- Even if the visualization window is constrained to have size h, the drawing algorithm that computes the drawing of the visualization window at instant t can know a subtree of T larger than $T_h(t)$. This subtree is denoted as $T_{h+k}(t)$. It has root $r(t)$ and is induced by the vertices of T at (oriented) distance at most $h+k$ from $r(t)$. Constant $k \geq 0$ is an integer that describes how much the algorithm is allowed to know about T more than what belongs to the visualization window. Tree $T_{h+k}(t)$ is called *knowledge window*; the *size* of the knowledge window is $h+k$.
- Very often, even if the graph is infinite, the visit of vertices follows an almost predictable pattern. One possibility for modeling this issue is to weight the children of each vertex with their probability to be visited after their parent. We make here the simpler assumption that for each vertex one *favorite child* is defined. Under this assumption the visualization path in any interval $[t_1, t_2]$ is $r(t_1), \ldots, r(t_2)$, where $r(t+1)$ $(t_1 \leq t < t_2)$ is always the favorite child of $r(t)$.

The drawing algorithms that we study are variants of the classical Reingold-Tilford algorithm [19]. This choice is motivated by the fact that the Reingold-Tilford algorithm is among the most robust, simple, effective, and well known algorithms of Graph Drawing.

Suppose to have a visualization window of size h and to have a knowledge window of size $h+k$. At instant t our algorithms compute a drawing of a tree larger than $T_h(t)$. Namely, they receive as input $T_{h+k}(t)$ and perform the following two steps:

- They prune $T_{h+k}(t)$ of the vertices that will never be part of any visualization window during the navigation. The resulting pruned tree is called $T_{h,k}(t)$ and is defined as $T_{h,k}(t) = \bigcup_{i=0}^{k} T_h(t+i)$.
- They compute a drawing of $T_{h,k}(t)$ and display the portion $T_h(t)$ of height h.

Note that the definition of tree $T_{h,k}(t)$ relies on the knowledge of the vertices $r(t), r(t+1), \ldots, r(t+k)$ of the visualization path. Also, observe that $T_{h,k}(t)$ has height $h+k$ and that $T_{h,0}(t) = T_h(t)$.

Figure 2 shows a portion of an infinite tree with a visualization window of height $h = 3$ and a knowledge window of height $h+k = 9$. In the figure, the black vertices belong to the visualization path and together with the grey vertices induce $T_{h,k}(t)$.

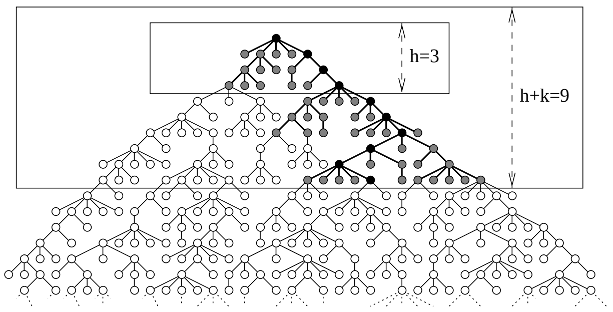


Figure 2: A portion of an infinite tree, a visualization window of height $h = 3$, and a knowledge window of height $h+k = 9$. A drawing algorithm computes a drawing of $T_{h,k}(t)$ (grey and black vertices). Only the portion of the drawing inside the visualization window is displayed on the screen. The portion of the tree induced by the white vertices is not taken into account by the drawing algorithm.

Our variants of the Reingold-Tilford Algorithms are characterized by a different choice for the embedding of $T_{h,k}(t)$.

`Leftmost-Embedding Algorithm` : The embedding of $T_{h,k}(t)$ is chosen so that for each vertex v along the visualization path, the favorite child of v is the leftmost child of v. As a result, the leftmost vertex at any given level of $T_{h,k}(t)$ is a vertex of the visualization path.

`Central-Embedding Algorithm` : The embedding of $T_{h,k}(t)$ is chosen so that for each vertex v along the visualization path, the favorite child of v is the middle child of v. A vertex u is the middle child of v if the number of vertices preceding u in the adjacency list of v differs by at most one from the number of vertices following u in the adjacency list of v.

`Random-Embedding Algorithm` : The embedding of $T_{h,k}(t)$ is chosen so that the favorite child of each vertex v along the visualization path can be in any position of the adjacency list of v.

We are interested in evaluating the performance of the above algorithms with respect to both dynamic and static quality measures. The *total-shift* dynamic quality measures the change of the x-value of a non-root vertex v of $T_{h-1}(t+1)$ with respect to its parent $p(v)$. Since our algorithms perform a pruning step before computing the drawing, we evaluate the total-shift dynamic quality measure with respect to pruned subtrees. Namely, we denote the total-shift dynamic quality measure as $m_D(h,k,t)$ and define it as follows:

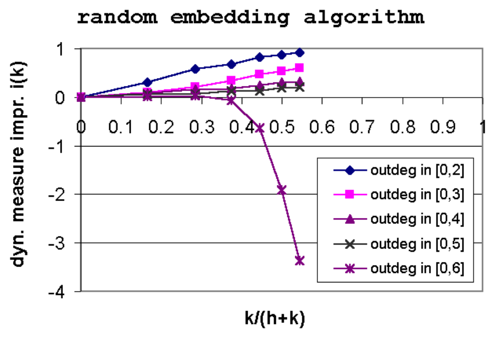
$$m_D(h,k,t) = \sum_{v \in T_{h-1}(t+1) - \{r(t+1)\}} |x(v,t) - x(p(v),t) - (x(v,t+1) - x(p(v),t+1))|.$$

In the definition, $x(v,t)$ is the x-coordinate of vertex v at instant t computed by an algorithm that considers $T_{h,k}(t)$.

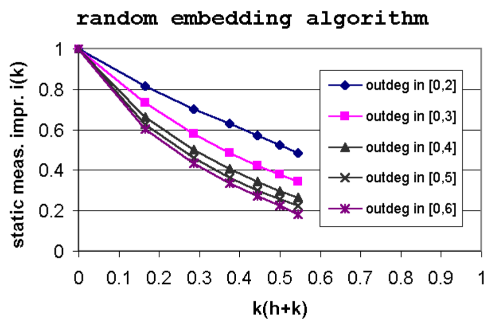
The static quality measure that we use to compare our algorithms evaluates the width of the drawing of the visualization window. More precisely, let $B_{h,k}(t)$ be the smallest isothetic box that contains the first h levels of a drawing of $T_{h,k}(t)$. We denote as $m_S(h,k,t)$ the width of box $B_{h,k}(t)$.

3 Experimental Results

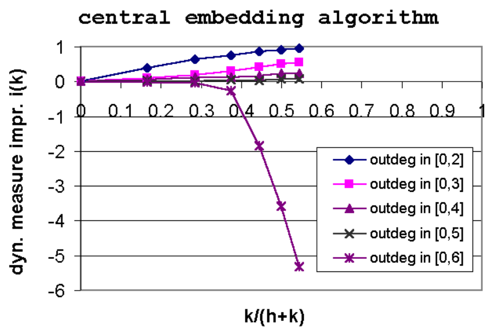
This section contains the results of an experimental comparison of `Random-Embedding Algorithm`, `Central-Embedding Algorithm`, and `Leftmost-Embedding Algorithm` with respect to the dynamic and static quality measures m_D and m_S introduced in Section 2.



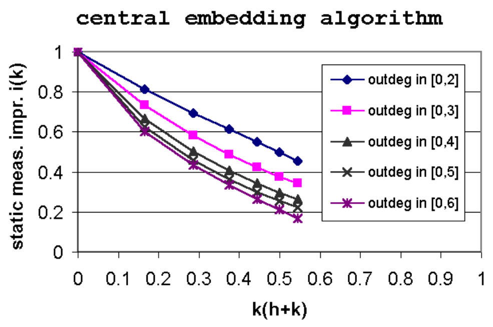
(a)



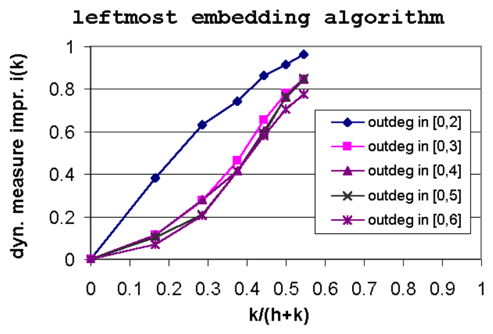
(b)



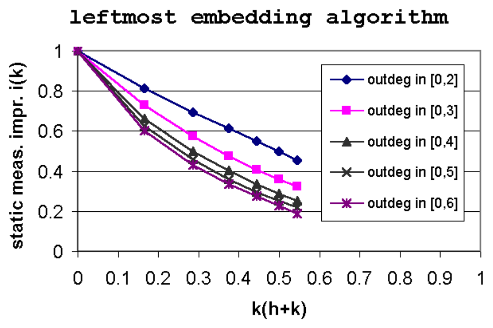
(c)



(d)



(e)



(f)

Figure 3: Experimental results for $h = 5$, $0 \leq k \leq 6$, minimum vertex outdegree 0, and maximum vertex outdegree between 2 and 5. On the left (right) the improvement or worsening of the dynamic (static) quality measure, for `Random`, `Central`, and `Leftmost Embedding Algorithm`, respectively, is shown.

In each experiment we consider a random path on a randomly generated tree. Since we actually perform a finite number l of navigation steps on each tree, once the visualization window height h and the knowledge k have been fixed, it suffices producing a tree with height $h+k+l$ to simulate the behavior of the algorithms on an infinite tree. Furthermore, in order to simulate a random infinite path on an infinite tree, we need to avoid choosing those random paths ending with a leaf before the expected length is reached.

The generation of a random tree and of the corresponding random path is performed as follows. In order to generate a tree of given height and with vertex outdegree ranging in a fixed interval, we randomly choose in such interval the number of children of each vertex, according to the uniform distribution, starting from the root and considering one level at a time, until we reach the desired tree height. Then we randomly select a leaf on the last level of the tree, implicitly defining a random path from the root to the chosen leaf.

In each experiment we fix the parameters h, k, and l, we randomly generate a tree of height $h+k+l$ and a path over it, and we perform l navigation steps, computing at each step the value of $m_D(h,k,t)$ and $m_S(h,k,t)$. For each tree, we average the values of the dynamic and static quality measures over the whole navigation and then we average these values over the entire set of randomly generated trees. The resulting measures are denoted as $m_D(h,k)$ for the total-shift dynamic quality measure and $m_S(h,k)$ for width static quality measure.

Instead of plotting the raw values of $m_D(h,k)$ and $m_S(h,k)$, we display their relative improvements, denoted as $i_D(h,k) = \frac{m_D(h,0)-m_D(h,k)}{m_D(h,0)}$ and $i_S(h,k) = \frac{m_S(h,0)}{m_S(h,k)}$, respectively.

Figure 3 shows the results of the experiments performed with $h = 5$, $0 \le k \le 6$, minimum vertex outdegree 0, and maximum vertex outdegree between 2 and 6. In particular, Figure 3.a, Figure 3.c, and Figure 3.e show the values of $i_D(h,k)$ for the `Random`, `Central`, and `Leftmost-Embedding Algorithm`, respectively, while Figure 3.b, Figure 3.d, and Figure 3.f show the corresponding values of $i_S(h,k)$. Each point on the diagrams is obtained computing the average over 1000 trees. The axes scales are chosen with the purpose of producing diagrams in the range $[0,1] \times [0,1]$.

Concerning the static quality measure, the behavior of $i_S(h,k)$ shows the expected worsening of the width of the drawing k increases.

For what concerns the dynamic quality measure, the diagrams show that the best values are obtained by `Leftmost-Embedding Algorithm` and the worst values by `Central-Embedding Algorithm`. In all charts, as the average outdegree of tree T increases, the dynamic improvement for any fixed value of k decreases. Also notice that $i_D(h,k)$ measured for `Random` and `Central-Embedding Algorithm` can be negative when the outdegree of the vertices increases (see Figure 3.a and Figure 3.b). This shows that not all strategies are able to take advantage of the knowledge of the future. On the other hand, $i_D(h,k)$ is always positive for `Leftmost-Embedding Algorithm`, which performs better than the other two in all experiments that we have run.

The charts relative to `Leftmost-Embedding Algorithm` illustrate tradeoffs between the values of the dynamic and static quality measures. In particular, as k increases, there is an improvement of the performances with respect to $m_D(h,k)$ and a worsening with respect to $m_S(h.k)$.

We can summarize the outcome of our experimental study as follows.

- Each level of knowledge identifies a tradeoff between the aesthetics of the drawing and the preservation of the mental map.
- An effective approach to the design of a drawing algorithm that achieves a pleasing compromise between aesthetic requirements and preservation of the mental map is based on using only a limited knowledge of the future.

4 Analysis

In this section we analytically compare the performances of `Leftmost-Embedding Algorithm` and of `Central-Embedding Algorithm`. This comparison considers the behavior of the total-shift quality measures for both algorithms when k goes to infinity. The analytical behavior of the curves of $m_D(h,k,t)$ for large values of k is consistent with the behavior observed experimentally for small values of k. We start by introducing some definitions and basic tools.

Let v be a vertex of $T_{h,k}(t)$ and consider a drawing Γ of $T_{h,k}(t)$. The difference between the x-coordinate of v and the x-coordinate of $p(v)$ in Γ is denoted as $\delta_{h,k}(v,t)$, i.e., $\delta_{h,k}(v,t) = x(p(v),t) - x(v,t)$. The following lemma shows the interplay between time and knowledge by relating the values of $\delta()$ at consecutive instants of time with the values of $\delta()$ for consecutive values of k.

Lemma 1 *Let v be a non-root vertex in $T_{h,k}(t) \cap T_{h,k}(t+1) = T_{h,k-1}(t+1)$. For any drawing produced by* `Leftmost-Embedding Algorithm` *(*`Central-Embedding Algorithm`*) we have $\delta_{h,k}(v,t) = \delta_{h,k-1}(v,t+1)$.*

Sketch of Proof: $\delta_{h,k}(v,t)$ is measured on the drawing of $T_{h,k}(t)$, while $\delta_{h,k-1}(v,t+1)$ is measured on the drawing of $T_{h,k-1}(t+1)$. Because $T_{h,k-1}(t+1)$ is the subtree of $T_{h,k}(t)$ rooted at $r(t+1)$, and since `Leftmost-Embedding Algorithm` (`Central-Embedding Algorithm`) draws a subtree with a bottom-up-strategy independent of the size of the enclosing supertree, we have that $\delta_{h,k}(v,t) = \delta_{h,k-1}(v,t+1)$. □

From the previous lemma we have that the variation of δ from instant t to instant $t+1$ is equal to the variation of δ when the knowledge at instant $t+1$ varies from $k-1$ to k.

Corollary 1 $\delta_{h,k}(v,t+1) - \delta_{h,k}(v,t) = \delta_{h,k}(v,t+1) - \delta_{h,k-1}(v,t+1)$.

Corollary 1 allows us to study the variation of δ by considering $T_{h,k-1}(t+1)$ and $T_{h,k}(t+1)$, instead of looking at $T_{h,k}(t)$ and $T_{h,k}(t+1)$. Since $T_{h,k-1}(t+1)$ and $T_{h,k}(t+1)$ share the root, we can univocally define a *level* for their vertices; we assume that the root has level 0.

4.1 Analysis of `Leftmost-Embedding Algorithm`

Consider a drawing Γ of $T_{h,k}(t+1)$ produced by `Leftmost-Embedding Algorithm`. All vertices that have the same level in $T_{h,k}(t+1)$ also have the same y-coordinate in Γ. Thus, we also talk about levels of Γ.

- Let $0 \le i \le k$ be a level of Γ, let $r(t+1+i)$ be the vertex of the visualization path at level i, and let u be its rightmost child. We denote with $\omega(i,k)$ the quantity $\omega(i,k) = x(u,t+1) - x(r(t+1+i),t+1)) = |\delta_{h,k}(u,t+1)|$. Also, we denote with $\mu(k)$ the quantity $\mu(k) = \omega(k,k)$.
- Let $1 \le i \le k$ be a level of Γ, let $r(t+i)$ be the vertex of the visualization path at level $i-1$, and let $r(t+1+i)$ its leftmost child. Also, let v be the rightmost child of $r(t+i)$ and let u be the rightmost child of $r(t+1+i)$. We denote with $\lambda(i,k)$ the quantity $\lambda(i,k) = x(v) - x(u)$.

Examples of the quantities $\lambda(i,k)$, $\omega(i,k)$, and $\mu(k)$ are depicted in Figure 4.

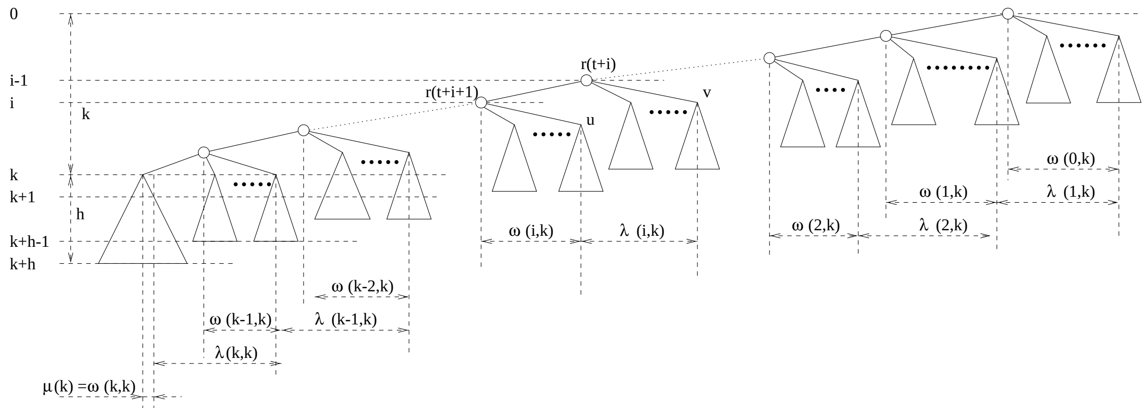


Figure 4: Quantities $\lambda(i,k)$, $\omega(i,k)$, and $\mu(k)$ for the analysis of the drawings produced by `Leftmost-Embedding Algorithm`.

Lemma 2 *For any positive integer k and for any integer $0 \le i \le k-1$, we have that $\lambda(i,k) = \lambda(i,k-1)$.*

Lemma 3 *Let t be an instant of time, and let h and k be the integers that define the size of the visualization window and of the knowledge window. For a drawing of $T_{h,k}(t)$ produced by* `Leftmost-Embedding Algorithm` *we have that we have that:*

$$\omega(i,k) = \frac{\mu(k)}{2^{k-i}} + \sum_{j=1}^{k-i} \frac{\lambda(i+j,k)}{2^j} \qquad i \in [0,k]$$

Sketch of Proof: The proof follows from the solution of the following recurrence equation (see also Figure 4):

$$\omega(i,k) = \begin{cases} \frac{1}{2}[\omega(i+1,k) + \lambda(i+j,k)] & 0 \leq i < k \\ \mu(k) & i = k \end{cases}$$

□

By means of Lemma 2 and Lemma 3 the following can be proved.

Lemma 4 *Let t be an instant of time, and let h and k be the integers that define the size of the visualization window and of the knowledge window. For a drawing of $T_{h,k}(t)$ produced by* `Leftmost-Embedding Algorithm` *we have that we have that:*

$$\omega(i,k) - \omega(i,k-1) = \frac{\lambda(k,k) - \mu(k) + 2\mu(k-1)}{2^{k-i}} \qquad i \in [0,k]$$

We are now ready to study the variation of ω as k goes to infinity. In order to ensure a finite limit, we need to bound the values of $\mu(k)$, $\mu(k-1)$, and $\lambda(k,k)$. Lemma 5 shows that this is always possible under the assumption that the maximum vertex degree of T is finite. The limit for the variation of ω as k goes to infinity is computed in Lemma 6.

Lemma 5 *Let T be an infinite tree whose maximum vertex outdegree is bounded by a constant. Then there exists a constant ξ such that $\forall k \ \lambda(k,k) - \mu(k) + 2\mu(k-1) \leq \xi$.*

Lemma 6 *Let T be an infinite tree whose maximum vertex outdegree is bounded by a constant. Then,*

$$\forall i \in [0,k] \qquad \lim_{k\to\infty} (\omega(i,k) - \omega(i,k-1)) = 0$$

Sketch of Proof: In view of Lemma 5, a constant ξ exists such that $\forall k \ \lambda(k,k) - \mu(k) + 2\mu(k-1) \leq \xi$. The claim immediately follows from Lemma 4:

$$\lim_{k\to\infty} (\omega(i,k) - \omega(i,k-1)) = \lim_{k\to\infty} \frac{\lambda(k,k) - \mu(k) + 2\mu(k-1)}{2^{k-i}} \leq \lim_{k\to\infty} \frac{\xi}{2^{k-i}} = 0$$

□

The following lemma provides a concise formula for the sum of the contributes to the dynamic quality measure $m_D(h,k,t)$ due to the vertices laying on the same level of $T_{h,k}(t)$.

Lemma 7 *Let t be an instant of time, and let h and k be the integers that define the size of the visualization window and of the knowledge window. For a drawing of $T_{h,k}(t)$ produced by* `Leftmost-Embedding Algorithm`, *we have that:*

$$\forall i \in [0,h-2] \qquad \sum_{\substack{v \,\in\, T_{h-1}(t+1) \\ \text{at level } i+1}} |\delta_{h,k-1}(v,t+1) - \delta_{h,k}(v,t+1)| = outdeg(r_i)(\omega(i,k) - \omega(i,k-1))$$

We are now able to express the dynamic quality measure $m_D(h,k,t)$ as a function of $\omega(i,k)$ and to compute the limit of the improvement $i_D(h,k,t) = \frac{m_D(h,0,t) - m_D(h,k,t)}{m_D(h,0,t)}$ as k goes to infinity.

Theorem 1 *Let t be an instant of time, and let h and k be the integers that define the size of the visualization window and of the knowledge window. For a drawing of $T_{h,k}(t)$ produced by* `Leftmost-Embedding Algorithm`, *let $m_D(h,k,t)$ be the total-shift dynamic quality measure. Then*

$$\lim_{k\to\infty} \frac{m_D(h,0,t) - m_D(h,k,t)}{m_D(h,0,t)} = 1$$

Sketch of Proof: We recall from Section 2 the definition of the dynamic quality measure m_D:

$$m_D(h,k,t) = \sum_{v \in T_{h-1}(t+1) - \{r(t+1)\}} |x(p(v),t) - x(v,t) - (x(p(v),t+1) - x(v,t+1))|$$

According to the definition of δ we have that $x(p(v),t) - x(v,t) = \delta_{h,k}(v,t)$ and $x(p(v),t+1) - x(v,t+1) = \delta_{h,k}(v,t+1)$ and that, in view of Lemma 1, $\delta_{h,k}(v,t) = \delta_{h,k-1}(v,t+1)$. Hence

$$m_D(h,k,t) = \sum_{v \in T_{h-1}(t+1) - \{r(t+1)\}} |\delta_{h,k-1}(v,t+1) - \delta_{h,k}(v,t+1)|$$

Partitioning vertices in $T_{h-1}(t+1)$ according to their level we have

$$m_D(h,k,t) = \sum_{i=0}^{h-2} \sum_{\substack{v \in T_{h-1}(t+1) \\ \text{at level } i+1}} |\delta_{h,k-1}(v,t+1) - \delta_{h,k}(v,t+1)|$$

which, in view of Lemma 7, and supposing $k \geq h$ in order to use ω in its domain, gives

$$m_D(h,k,t) = \sum_{i=0}^{h-2} outdeg(r(t+1+i))|\omega(i,k) - \omega(i,k-1)|$$

Therefore, the truth of the statement follows from Lemma 6. □

4.2 Analysis of `Central-Embedding Algorithm`

In this section we compare the behavior of `Central-Embedding Algorithm` to that of `Leftmost-Embedding Algorithm` with respect to the total-shift dynamic quality measure.

Consider a drawing of $T_{h,k}(t+1)$ produced by `Central-Embedding Algorithm`. The quantities $\omega(i,k)$ and $\mu(k)$ can be defined in the same way as in Subsection 4.1. We need two new definitions. Refer also to Figure 5.

Let $r(t+1+i)$ be a vertex at level $1 \leq i \leq k$ in the visualization path and let $p(r(t+1+i))$ be its parent. Let u and v be the rightmost and the leftmost children of $r(t+1+i)$, respectively. Let w and z be the rightmost and the leftmost children of $p(r(t+1+i))$, respectively. We denote with $\lambda^l(i,k)$ the quantity $\lambda^l(i,k) = x(v) - x(z)$ and with $\lambda^r(i,k)$ the quantity $\lambda^r(i,k) = x(w) - x(u)$.

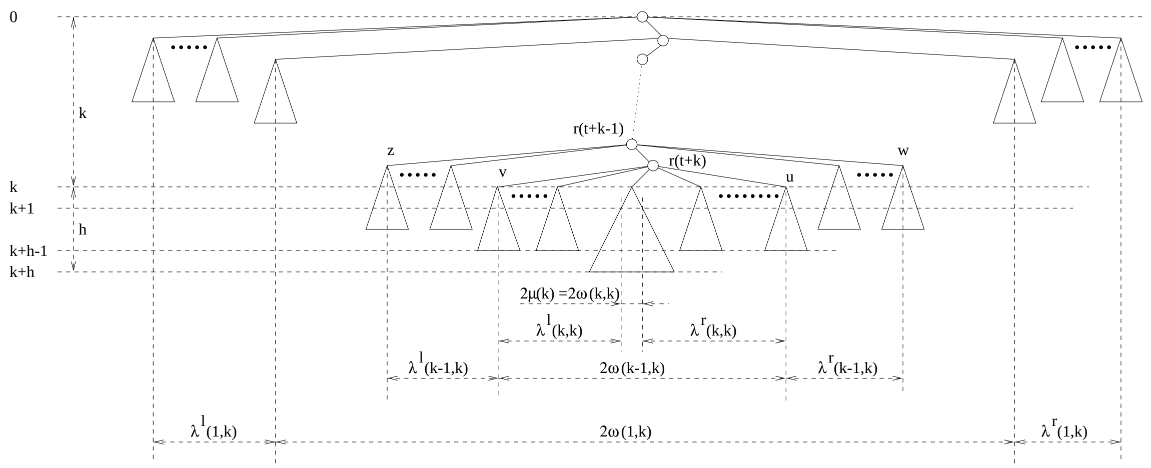


Figure 5: Pruned tree and quantities $\lambda^l(i,k)$, $\lambda^r(i,k)$, $\omega(i,k)$, and $\mu(k)$ for the analysis of the drawings produced by `Central-Embedding Algorithm`.

With reasoning similar to that of Lemma 3 and Lemma 7, the following lemmas can be proved.

Lemma 8 *Let t be an instant of time, and let h and k be the integers that define the size of the visualization window and of the knowledge window. For a drawing of $T_{h,k}(t)$ produced by* `Central-Embedding Algorithm` *we have that:*

$$\omega(i,k) = \mu(k) + \sum_{j=1}^{k-i} \frac{\lambda^l(i+j,k) + \lambda^l(i+j,k)}{2} \qquad i \in [0,k]$$

Lemma 9 *Let t be an instant of time, and let h and k be the integers that define the size of the visualization window and of the knowledge window. For a drawing of $T_{h,k}(t)$ produced by* `Central-Embedding Algorithm` *we have that:*

$$\forall i \in [0, h-2] \qquad \sum_{\substack{v \in T_{h-1}(t+1) \\ at\ level\ i+1}} |\delta_{h,k-1}(v,t+1) - \delta_{h,k}(v,t+1)| = [outdeg(r_i)-1](\omega(i,k)-\omega(i,k-1))$$

By means of Lemma 7 and Lemma 9, we can prove the following theorem, which states that for any sufficiently large value of k the performances of `Leftmost-Embedding Algorithm` are always better than the performances of `Central-Embedding Algorithm` .

Theorem 2 *Let t be an instant of time, and let h and k be the integers that define the size of the visualization window and of the knowledge window. Let ${m_D}^l(h,k,t)$ and ${m_D}^c(h,k,t)$ be the total-shift dynamic quality measure of* `Leftmost-Embedding Algorithm` *and* `Central-Embedding Algorithm`*, respectively. Then*

$$\exists k_0 \geq 0 : \quad \forall k \geq k_0 \qquad {m_D}^c(h,k,t) \geq {m_D}^l(h,k,t)$$

5 Conclusions and Open Problems

This paper has proposed a new framework for visually supporting the navigation of infinite graphs in which the knowledge of part of the future moves of the user can be exploited to preserve her mental map. As an illustration of this framework, the problem of drawing the portions of an infinite tree that the user visits during her navigation has been studied. Our research seems to open several new problems.

We mention here some of those that in our opinion are among the most relevant.

1. Devise new strategies for supporting the navigation of infinite trees.
2. Extend our study to different families of graphs, for example planar graphs.
3. The model in which every vertex v of an infinite graph has one favorite neighbor can be extended by assuming that each neighbor of v has a certain probability to be visited by the user in her next navigation step.

References

[1] *Astra.* Mercury international. `http://www.merc-int.com/`.

[2] U. Brandes and D. Wagner. A bayesian paradigm for dynamic graph layout. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes Comput. Sci.*, pages 236–247. Springer-Verlag, 1998.

[3] R. F. Cohen, G. Di Battista, R. Tamassia, and I. G. Tollis. Dynamic graph drawings: Trees, series-parallel digraphs, and planar *ST*-digraphs. *SIAM J. Comput.*, 24(5):970–1001, 1995.

[4] P. Crescenzi, C. Demetrescu, I. Finocchi, and R. Petreschi. Leonardo: a software visualization system. In G. Italiano and S. Orlando, editors, *Workshop on Algorithm Engineering*, pages 146–155, 1997.

[5] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing.* Prentice Hall, Upper Saddle River, NJ, 1999.

[6] C. A. Duncan, M. Goodrich, and S. Kobourov. Balanced aspect ratio trees and their use for drawing very large graphs. In S. Whitesides, editor, *Graph Drawing (Proc. GD '98)*, volume 1547 of *Lecture Notes Comput. Sci.*, pages 384–393. Springer-Verlag, 1999.

[7] P. Eades, R. F. Cohen, and M. L. Huang. Online animated graph drawing for web navigation. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes Comput. Sci.*, pages 330–335. Springer-Verlag, 1998.

[8] P. Eades, W. Lai, K. Misue, and K. Sugiyama. Preserving the mental map of a diagram. In *Proceedings of Compugraphics 91*, pages 24–33, 1991.

[9] T. R. Henry and S. E. Hudson. Viewing large graphs. Technical Report 90-13, Department of Computer Science, University of Arizona, 1990.

[10] *Hyperbolic Tree.* Inxight. `http://www.hyperbolictree.com/`.

[11] K. Kaugars, J. Reinfelds, and A. Brazma. A simple algorithm for drawing large graphs on small screens. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes Comput. Sci.*, pages 278–281. Springer-Verlag, 1995.

[12] D. Kimelman, B. Leban, T. Roth, and D. Zernik. Reduction of visual complexity in dynamic graphs. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes Comput. Sci.*, pages 218–225. Springer-Verlag, 1995.

[13] E. B. Messinger, L. A. Rowe, and R. H. Henry. A divide-and-conquer algorithm for the automatic layout of large directed graphs. *IEEE Trans. Syst. Man Cybern.*, SMC-21(1):1–12, 1991.

[14] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *J. Visual Lang. Comput.*, 6(2):183–210, 1995.

[15] S. Moen. Drawing dynamic trees. *IEEE Softw.*, 7:21–28, 1990.

[16] T. Munzner. Exploring large graphs in 3d hyperbolic space. *Comp. Graphics and its Applications*, 8(4):18–23, 1998.

[17] T. Munzner. Drawing large graphs with h3viewer and site manager. In S. Whitesides, editor, *Graph Drawing (Proc. GD '98)*, volume 1547 of *Lecture Notes Comput. Sci.*, pages 384–393. Springer-Verlag, 1999.

[18] S. North. Incremental layout in DynaDAG. In F. J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes Comput. Sci.*, pages 409–418. Springer-Verlag, 1996.

[19] E. Reingold and J. Tilford. Tidier drawing of trees. *IEEE Trans. Softw. Eng.*, SE-7(2):223–228, 1981.

[20] G. J. Wills. NicheWorks - interactive visualization of very large graphs. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes Comput. Sci.*, pages 403–414. Springer-Verlag, 1998.