Information Processing Letters 112 (2012) 418–422

Contents lists available at SciVerse ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Drawing trees in a streaming model ☆

Carla Binucci [a], Ulrik Brandes [b], Giuseppe Di Battista [c], Walter Didimo [a,*], Marco Gaertler [d], Pietro Palladino [e], Maurizio Patrignani [c,**], Antonios Symvonis [f], Katharina Zweig [g]

[a] Dipartimento di Ing. Elettronica e dell'Informazione, Università degli Studi di Perugia, Italy
[b] Department of Computer and Information Science, University of Konstanz, Germany
[c] Dipartimento di Informatica e Automazione, Università Roma Tre, Italy
[d] Institute of Theoretical Computer Science, University of Karlsruhe, Germany
[e] Dipartimento di Medicina Sperimentale e Scienze Biochimiche, Università degli Studi di Perugia, Italy
[f] Department of Mathematics, National Technical University of Athens, Greece
[g] Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg, Germany

ARTICLE INFO

Article history:
Received 29 September 2011
Received in revised form 10 January 2012
Accepted 14 February 2012
Available online 28 February 2012
Communicated by R. Uehara

Keywords:
Design of algorithms
Graph algorithms
Online algorithms
Graph drawing
Streaming
Large graphs

ABSTRACT

We pose a new visualization challenge, asking Graph Drawing algorithms to cope with the requirements of Streaming applications. In this model a source produces a graph one edge at a time. When an edge is produced, it is immediately drawn and its placement cannot be altered. The drawing has an image persistence, that controls the lifetime of edges. If the persistence is k, an edge remains in the drawing for the time spent by the source to generate k edges, and then it fades away. In this model we study the area requirement of planar straight-line grid drawings of trees and we assess the output quality of the presented algorithms by computing the competitive ratio with respect to the best known offline algorithms.

1. Introduction

We consider the following model. A source produces a graph one edge at a time. When an edge is produced, it is immediately drawn (i.e., before the next edge is produced) and its drawing cannot be altered. The drawing has an image persistence, that controls the lifetime of edges. If the persistence is k, an edge remains in the drawing for the time spent by the source to generate k edges, and then it fades away.

Studying this model, which we call *streamed graph drawing*, is motivated by the challenge of offering visualization facilities to streaming applications, where massive amounts of data, too large even to be stored, are produced and processed at a very high rate [2]. The data are available one element at a time and need to be processed quickly and with limited resources. Examples of application fields include computer network traffic analysis, logging of security data, stock exchange quotes' correlation, etc.

For the user of the visualization facility it is natural to associate any graphic change with a new datum coming from the stream. Hence, moving pieces of the drawing

☆ Work on this problem began at the BICI Workshop on Graph Drawing: Visualization of Large Graphs, held in Bertinoro, Italy, in March 2008. Work supported in part by the MIUR project AlgoDEEP prot. 2008TFBWL4. Part of the research was conducted in the framework of ESF project 10-EuroGIGA-OP-003 GraDR "Graph Drawings and Representations". An extended abstract of this paper appeared in the proceedings of the 17th International Symposium on Graph Drawing, GD 2009 (Binucci et al., 2009 [1]).

* Principal corresponding author.
** Corresponding author.
E-mail addresses: binucci@diei.unipg.it (C. Binucci), Ulrik.Brandes@uni-konstanz.de (U. Brandes), gdb@dia.uniroma3.it (G. Di Battista), didimo@diei.unipg.it (W. Didimo), marco.gaertler@kit.edu (M. Gaertler), pietropalladino@gmail.com (P. Palladino), patrigna@dia.uniroma3.it (M. Patrignani), symvonis@math.ntua.gr (A. Symvonis), katharina.zweig@iwr.uni-heidelberg.de (K. Zweig).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2012.02.011

would be potentially ambiguous. On the other hand, the drawing should have a size as small as possible.

Although streamed graph drawing is related to incremental and dynamic graph drawing, it is qualitatively different from both. In incremental graph drawing the layout is constructed step by step according to a precomputed vertex ordering that ensures invariants regarding, e.g., its shape [3,4]. In streamed graph drawing the order cannot be chosen. Dynamic graph drawing [5–7] usually refers to drawing sequences of graphs, where drawings of consecutive graphs should be similar. Insertions and/or deletions of vertices/edges are allowed and the current graph must be drawn without knowledge of future updates. However, the current layout is only weakly constrained by previous drawings. In streamed graph drawing modifications concern only single edges and previous layout decisions may not be altered.

While there is some work on computing properties of streamed graphs (see, e.g., [8–10]), as far as we know this is the first time that the problem of drawing the k most recent edges of a stream has been addressed.

In this paper, we concentrate on trees and we make some assumption on the ordering in which the edges of the tree are visited. Namely, we consider the area requirement for planar straight-line grid drawings of trees, and we assume that the edges are streamed corresponding to an Eulerian tour of the tree. A typical real-world scenario in which this kind of streamed trees occur is the live representation of procedure call trees in dynamic program analysis. Each procedure may call other procedures and each call suspends the calling procedure until the called procedure has terminated. Note that, even medium size programs may have billions of procedure calls during a single run, which motivates the design of visualization tools for trace exploration [11–14]. Also, drawing a graph in a small area is a typical goal in graph visualization (see, e.g., [15]).

Since a streamed graph drawing algorithm is a special case of an online algorithm, it is reasonable to assess its output quality in terms of its competitive ratio with respect to the best known offline algorithm.

This paper is organized as follows. In Section 2 we introduce the concept of streamed graph drawing. Area requirements for tree drawings are derived in Section 3, and we conclude with directions for future work in Section 4.

2. Framework

Let $G = (V, E)$ be a simple undirected graph. A *straight-line grid drawing* $\Gamma = \Gamma(G)$ is a geometric representation of G such that each vertex is drawn as a distinct point of an integer-coordinate grid, and each edge is drawn as a straight-line segment between the points associated with its end-vertices. A drawing is *planar* if no two edges cross. Since we only consider planar straight-line grid drawings we simply refer to them as *drawings* in the remainder.

Given a subset of edges $E' \subseteq E$, the *edge-induced* (*sub*)*graph* $G[E']$ contains exactly those vertices of V incident with edges in E', and the edges in E'. We study the problem of drawing a (potentially infinite) graph G described by a sequence of edges $(e_1, e_2, e_3, \ldots)$, which we call a *stream of edges*, where e_i is known at time i. Throughout this paper, let $W_i^k = \{e_{i-k+1}, \ldots, e_i\}$ denote a *window* of the stream of size k and let $E_i = \{e_1, \ldots, e_i\}$ denote the *prefix* of the stream of length i. Observe that $E_i = W_i^i$.

Our goal is to design online drawing algorithms for streamed graphs. An online drawing algorithm incrementally constructs a drawing of the graph, by adding one edge at a time according to the order in which they appear in the stream. Once a vertex is placed, however, its placement cannot be altered until the vertex is removed.

More formally, we address the following problem: Let Γ_0 be an initially empty drawing. At each time $i \geqslant 1$ and for some fixed parameter $k \geqslant 1$, called *persistence*, determine a drawing Γ_i of $G_i = G[W_i^k]$ by adding e_i to Γ_{i-1} and dropping (if $i > k$) e_{i-k} from Γ_i.

We assume that our memory is bounded by $O(k)$.

Since streamed graph drawing algorithms are special online algorithms, an important assessment of quality is their competitive ratio. For a given online drawing algorithm A and some measure of quality, consider any stream of edges $S = (e_1, e_2, \ldots)$. Denote by $A(S)$ the quality of A executed on S, and by $Opt(S)$ the quality achievable by an optimal offline algorithm, i.e. an algorithm that knows the streaming order in advance. Where possible, we measure the effectiveness of A by evaluating its *competitive ratio*: $R_A = \max_S \frac{A(S)}{Opt(S)}$.

In the remainder of the paper we restrict our attention to the case where G is a tree, the goal is to determine a planar straight-line grid drawing, and the measure of quality is the *area* required by the drawing, i.e., the number of grid points contained in the minimum bounding box for the drawing. We recall that, static algorithms to draw an n-vertex tree in $\Theta(n)$ area are known if the tree is a binary tree [16] or if its vertex-degree is bounded by $\sqrt{n}$ [17]. The best known area bound for general trees is $O(n \log n)$ [18,19].

3. Drawing a streamed tree

We consider the following scenario, corresponding to the intuitive notion of a user traversing an undirected tree: the edges of the stream are given according to an Eulerian tour of the tree where we suppose that the persistence k is much smaller than the number of the edges of the tree (the tree may be considered "infinite"). Each (undirected) edge (u, v) is traversed exactly twice, once from u to v and once from v to u: the direction in which the edge is traversed for the first time is called the *forward* direction; the other direction is called the *backward* direction.

This corresponds to a DFS traversal where backtracks explicitly appear. Observe that window W_i^k contains in general both forward and backward edges and that $G[W_i^k]$ is always connected. Fig. 1 shows an example of an Eulerian tour where several windows of size 5 are highlighted: window W_1 contains two forward and three backward edges; window W_5 contains all backward edges (in the figure, the DFS visit starts from the rightmost vertex of the drawing and proceeds counter-clockwise).

In this scenario a vertex may be encountered several times during the traversal. Consider edge $e_i = (u, v)$ and assume that the Eulerian tour moves from u to v. We