*Erschienen in: Information Processing Letters ; 112 (2012), 11. - S. 418-422*
*https://dx.doi.org/10.1016/j.ipl.2012.02.011*

# Drawing Trees in a Streaming Model[⋆]

Carla Binucci[1], Ulrik Brandes[2], Giuseppe Di Battista[3],
Walter Didimo[1], Marco Gaertler[4], Pietro Palladino[1],
Maurizio Patrignani[3], Antonios Symvonis[5], and Katharina Zweig[6]

[1] Dipartimento di Ing. Elettronica e dell'Informazione, Università degli Studi di Perugia
[2] Fachbereich Informatik & Informationswissenschaft, Universität Konstanz
[3] Dipartimento di Informatica e Automazione, Università Roma Tre
[4] Fakultät für Informatik, Universität Karlsruhe (TH)
[5] Department of Mathematics, National Technical University of Athens
[6] Department of Biological Physics, Eötvös Loránd University

**Abstract.** We introduce a data stream model of computation for Graph Drawing, where a source produces a graph one edge at a time. When an edge is produced, it is immediately drawn and its drawing can not 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, then it fades away. In this model we study the area requirement of planar straight-line grid drawings of trees, with different streaming orders, layout models, and quality criteria. 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 can not be altered. The drawing has an image persistence, that controls the lifetime of edges. If the persistence is infinite, edges are never removed from the drawing. Otherwise, suppose 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 [12]. 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 would be potentially ambiguous. On the other hand, the drawing should have a size as limited as possible.

[⋆] Work on this problem began at the BICI Workshop on Graph Drawing: Visualization of Large Graphs, held in Bertinoro, Italy, in March 2008.

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 [3,7]. In streamed graph drawing the order cannot be chosen. Dynamic graph drawing [4,11,13] 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., [1,5,8]), very little has been done in the context of graph drawing. A result that applies to streamed graph drawing with infinite persistence is shown in [13] in what is called *no change scenario*. In that paper, a graph of maximum degree four is available one-vertex-at-time and it is drawn orthogonally and with a few crossings.

We consider both a finite persistence and an infinite persistence model. Our results in these models concern the area requirement for planar straight-line grid drawings of trees, where we assume that the tree is streamed in such a way that the subtree to be drawn is connected. 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. The obtained results are summarized in Table 1, where $n$ is the number of vertices of the current graph. For each type of streaming order and for each class of trees investigated within each model, the table reports the competitive ratio of a (specific) drawing algorithm, and the corresponding lemma/theorem. The table puts in evidence the practical applicability of the finite persistence model. On the other hand, the results on the infinite persistence model show the intrinsic difficulty of the problem. In fact, in the paper we prove that a large family of algorithms for the infinite persistence model requires $\Omega(2^{\frac{n}{8}}/n)$ competitive ratio to draw binary trees (see Lemma 5).

Another way to interpret our results on the infinite persistence model is the following: All the area-efficient tree-drawing algorithms known in the literature have the capability to inspect the entire tree for exploiting some balancing consideration. In the infinite persistence model we ask the question of which is the achievable area bound if such an inspection can not be done.

This paper is organized as follows. In Sect. 2 we introduce the concept of streamed graph drawing. Area requirements for tree drawings in our two main models are derived in Sects. 3 and 4, and we conclude with directions for future work in Sect. 5.

## 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.

**Table 1.** Summary of the results: competitive ratios of the proposed algorithms

**Finite persistence model** (constant persistence $k$)

| Streaming order | Graph class | Area (competitive ratio) | |
|---|---|---|---|
| Eulerian tour | tree | $O(k^2)$ | Sect. 3, Theorem 2 |

**Infinite persistence model** (unbounded memory, $n$ is the current graph size)

| Streaming order | Graph class | Area (competitive ratio) | |
|---|---|---|---|
| connected | binary tree | $\Theta(2^n)$ | Sect. 4.1, Lemma 4 |
| | tree, max. degree $d$ | $\Theta((d-1)^n)$ | Sect. 4.1, Lemma 6 |
| | tree | $\Omega(2^n/n)$ | Sect. 4.1, Lemma 7 |
| BFS, DFS | tree | $\Theta(n)$ | Sect. 4.2, Lemma 8 |
| layered | tree, max. degree $d$ | $\Theta(dn)$ | Sect. 4.3, Lemma 9 |

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 $(e_1, e_2, e_3, \dots)$, which we call a *stream of edges*, where $e_i$ is known at time $i$. Throughout this paper, let $W_i^k = \{e_{i-k+1}, \dots, e_i\}$ denote a *window* of the stream of size $k$ and let $E_i = \{e_1, \dots, e_i\}$ 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, the decision must not be altered unless the vertex is removed.

Let $\Gamma_0$ be an initially empty drawing. We deal with two models.

**Finite persistence model.** At each time $i \geq 1$ and for some fixed parameter $k \geq 1$, called *persistence*, determine a drawing $\Gamma_i$ of $G_i = G[W_i^k]$ by adding $e_i$ to $\Gamma_{i-1}$ and dropping (if $i > k$) $e_{i-k}$ from $\Gamma_i$.

**Infinite persistence model.** At each time $i \geq 1$, determine a drawing $\Gamma_i$ of $G_i = G[E_i]$ by adding $e_i$ to $\Gamma_{i-1}$.

We relate the connectivity of the graph to the persistence of the drawing. If the persistence $k$ is finite, a stream of edges is *connected* if $G[W_i^k]$ is connected for all $i \geq 1$. If the persistence is infinite, then a stream is *connected* if $G[E_i]$ is connected for all $i \geq 1$. In both models we assume that the stream of edges is connected. Also, in the finite persistence model 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 $S = (e_1, e_2, \dots)$. 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.

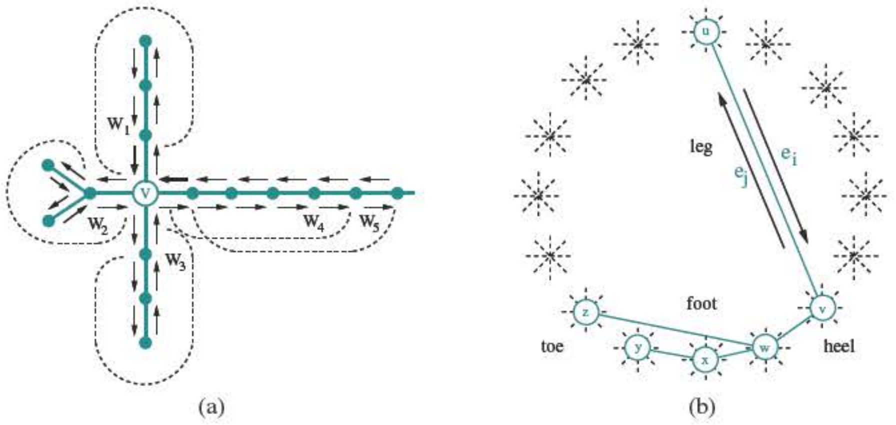

**Fig. 1.** (a) An Eulerian tour with a persistence $k = 5$. When $W_5$ is the current window, vertex $v$ is disappeared from the drawing. (b) A leg of vertex $u$.

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. Note that, $Opt(S) = \Theta(n)$ if $G[S]$ is a binary tree with $n$ vertices [9] or if G[S] is a tree with $n$ vertices and vertex degree bounded by $\sqrt{n}$ [10]. The best known area bound for general trees is $O(n \log n)$ [6,14]. In the next two sections we give upper and lower bounds on the area competitive ratio of streamed tree drawing algorithms under several streaming orders.

## 3 Finite Persistence Drawings of Trees

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 edge is traversed exactly twice: the first time in the *forward* direction and the second time in 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. Figure 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 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 say that $e_i$ *leaves* $u$ and *reaches* $v$. Also, if $v$ was already a vertex of $G_{i-1}$ (and hence is already drawn in $\Gamma_{i-1}$) then, we say that $e_i$ *returns* to $v$. Otherwise, $v$ has to be inserted into the drawing $\Gamma_i$ of $G_i$. Observe that if a vertex $v$, reached at time $i$, is reached again at time $j$, with $j > i + k + 1$, and is not reached at any intermediate time, then $v$ has (in general) two different representations in $\Gamma_i$ and $\Gamma_j$.

The first algorithm presented in this section is the following. Consider $m$ integer-coordinate points $p_0, p_1, \dots, p_{m-1}$ in convex position. An easy strategy is to use such points clockwise in a greedy way. At each time $i$, we maintain an index $next_i$ such that point $p_{next_i}$ is the first unused point in clockwise order. The first edge $e_1$ is drawn between points $p_0$ and $p_1$ and $next_2 = 2$. Suppose that edge $e_i = (u, v)$ has to be added to the drawing. If $v$ is not present in $\Gamma_{i-1}$, assign to $v$ the coordinates of $p_{next_i}$ and set $next_{i+1} = (next_i + 1) \mod m$. We call this algorithm Greedy-Clockwise (GC).

Algorithm GC guarantees a non-intersecting drawing provided that two conditions are satisfied for all $i$: (**Condition 1**) Point $p_{next_i}$ is not used in $\Gamma_i$ by any vertex different from $v$. (**Condition 2**) Edge $e_i$ does not cross any edge of $\Gamma_i$. Lemma 1 and Lemma 2 show that satisfying Condition 1 implies satisfying Condition 2. Let $w$ be a vertex of $\Gamma_i$, we denote by $i(w)$ the time when vertex $w$ entered $\Gamma_i$.

**Lemma 1.** *Let $\Gamma_i$ be a drawing of $G_i$ constructed by Algorithm* GC *and let $v_1$, $v_2$, and $v_3$ be three vertices of $G_i$ such that $i(v_1) < i(v_2) < i(v_3)$ in $\Gamma_i$. If there is a sequence of forward edges from $v_1$ to $v_3$, then there is a sequence of forward edges from $v_1$ to $v_2$.*

*Proof.* Consider edges $e_{i(v_1)} = (v_0, v_1)$ and $e_j = (v_1, v_0)$ of the stream. The Eulerian tour implies that the vertices reached by a forward path from $v_1$ are those vertices incident to some edge $e_h$, with $i(v_1) < h < j$. Suppose for a contradiction that $v_2$ is not reached by a forward path from $v_1$. Since $v_2$ was drawn after $v_1$, this implies $i(v_2) > j$. It follows that also $i(v_3) > j$. Hence, $v_3$ can not be reached by a forward path from $v_1$. □

**Lemma 2.** *Let $\Gamma_{i-1}$ be a drawing of $G_{i-1}$ constructed by Algorithm* GC *and consider a vertex $v$ that is not in $G_{i-1}$ and should be added to $G_{i-1}$ at time $i$. If Condition 1 is satisfied, then no crossing is introduced by drawing $v$ at $p_{next_i}$.*

*Proof.* Let $e_i = (u, v)$. Draw $v$ on $p_{next_i}$. Since Condition 1 is satisfied, then $p_{next_i}$ is not used by any vertex. Suppose for contradiction that $\Gamma_i$ has a crossing. It follows that there exists in $\Gamma_i$ an edge $(x, y)$, such that vertices $x$, $u$, $y$, $v$ appear in this relative order in the clockwise direction. By Condition 1 and since the points are used in a greedy way, $i(x) < i(u) < i(y) < i(v)$. Because of edge $(x, y)$, there is a forward path from $x$ to $y$ and hence by Lemma 1 there is a forward path from $x$ to $u$. Analogously, because of edge $(u, v)$, there is a forward path from $u$ to $v$ and hence by Lemma 1 there is a forward path from $u$ to $y$. Hence, there is an undirected cycle in $G_i$ involving $x$, $u$, and $y$. This is a contradiction since we are exploring a tree. □

Consider two edges $e_i = (u, v)$ and $e_j = (v, u)$, with $j > i$. Observe that $j - i$ is odd. Edges $e_i, e_{i+1}, \dots, e_j$ are a *leg* of $u$. Vertices discovered at times $i, i+1, \dots, j$, i.e., the $\frac{j-i+1}{2}$ distinct vertices incident to edges $e_{i+1}, \dots, e_{j-1}$, are a *foot* of $u$. Node $v$ is the *heel* of the foot and the last discovered vertex of the foot is the *toe*. Figure 1(b) shows the drawing of a leg (and provides a hint of why its vertices are called a foot). A foot is itself composed of smaller feet, where the smallest possible foot is when a leaf of the tree is reached, that is, when its heel and its toe are the same vertex (as for vertex $y$ of Fig. 1(b)).

Consider the case when $j - i \leq k$. This implies that $u$ is present in all the drawings $\Gamma_{i-1}, \ldots, \Gamma_{j+k}$. In this case we say that the foot is a *regular foot* (or *R-foot*). Otherwise, we say that it is an *extra-large foot* (or *XL-foot*).

*Property 1.* A regular foot has maximum size $\lceil \frac{k}{2} \rceil$.

Observe that in any drawing constructed by Algorithm GC the vertices of a regular foot are contiguously placed after its heel, the toe being the last in clockwise order.

*Property 2.* Let $i$ be the time when an extra-large foot of $v$ is entered by the Eulerian tour. Vertex $v$ disappears from the drawing at time $i + k$.

Now, we exploit the above properties and lemmas to prove that, if $k$ is the persistence of the drawing and if the tree has maximum degree $d$ (where a binary tree has $d = 3$), then it suffices using $\lceil \frac{k}{2} \rceil \cdot (d-1) + k + 1$ points in convex position to guarantee to GC that Condition 1 is satisfied. In order to prove this we need the following lemma.

**Lemma 3.** *Consider Algorithm* GC *on $m$ points in convex position. Suppose that for each vertex $v$ it holds that during the time elapsing from when $v$ is discovered and when it disappears from the drawing at most $m - 1$ other vertices are discovered. Then Condition 1 holds at each time.*

*Proof.* Suppose, for a contradiction that there exists a vertex $u$, discovered at time $i$, for which Condition 1 does not hold because point $p_{next_i}$ is used by vertex $w \neq u$. Since GC is greedy, after $u$ has been inserted all the $m$ points have been used. This implies that after $w$ and before $u$, $m - 1$ vertices have been discovered. Summing up, we have that $w$ violates the condition of the statement.

**Theorem 1.** *Let $S$ be a stream of edges produced by an Eulerian tour of a tree of degree at most $d$. Algorithm* GC *draws $S$ with persistence $k$ without crossings on $\lceil \frac{k}{2} \rceil \cdot (d - 1) + k + 1$ points in convex position. Also $R_{GC} = O(d^3 k^2)$.*

*Proof.* Due to Lemma 2 it suffices to show that Condition 1 holds at each time $i$. We exploit Lemma 3 to show that during the time elapsing from when a vertex $v$ is discovered and when it disappears from the drawing at most $\lceil \frac{k}{2} \rceil \cdot (d - 1) + k$ other vertices are discovered. Suppose $v$ is discovered by edge $e_i = (u, v)$. Three cases are possible: (i) $v$ is a leaf; (ii) all feet of $v$ are regular; (iii) $v$ has an XL-foot. Case (i) is simple: we have that $v$ disappears from the drawing at time $i + k + 1$. Hence, at most $k$ vertices can be discovered before it disappears. In Case (ii) since each R-foot can have at most $\lceil \frac{k}{2} \rceil$ vertices (Property 1) and since at most $(d - 1)$ of them can be traversed, the maximum number of vertices that can be discovered after $v$ enters the drawing and before it disappears is $\lceil \frac{k}{2} \rceil \cdot (d-1) + k$ (see Fig. 1(a) for an example with $k = 5$). In Case (iii), because of Property 2, after the XL-foot is entered, at most $k$ vertices can be discovered before $v$ disappears. Hence, the worst case is that the XL-foot follows $d - 2$ R-feet. Overall, a maximum of $\lceil \frac{k}{2} \rceil \cdot (d - 2) + k$ vertices can be discovered before $v$ disappears.

Regarding the competitive ratio, $m$ grid points in convex position take $O(m^3)$ area [2], and therefore the area of the drawing of our online algorithm is $\Theta(d^3 k^3)$. Finally, any offline algorithm requires $\Omega(k)$ area for placing $O(k)$ vertices. □

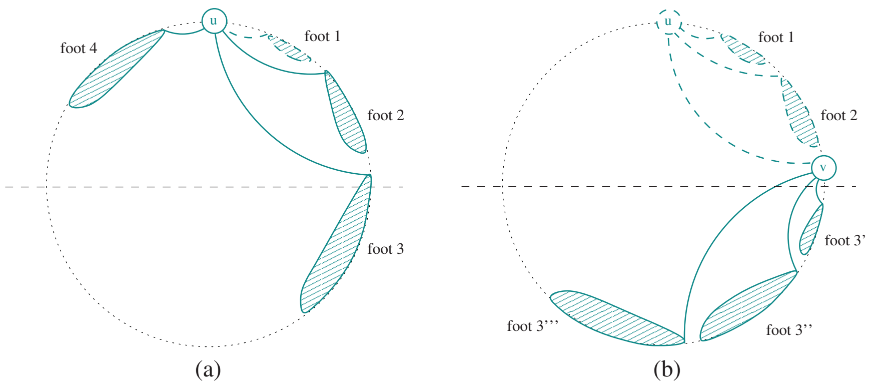

**Fig. 2.** (a) Feet 1, 2, and 3 are drawn by GC, foot 4 is drawn by GCC. (b) Foot 3 is an XL-foot of $u$. Its size is large enough to promote $v$ as the oldest vertex in place of $u$.

Theorem 1 uses a number of points that is proportional to the maximum degree of the tree. In the following we introduce a second algorithm that uses a number of points that only depends on the persistence $k$.

Intuitively, the basic strategy is to alternate Algorithm GC with its mirrored version, called GREEDY-COUNTER-CLOCKWISE (GCC), where, at each step, $next_i$ is possibly decreased rather than increased. Namely, let $old(\Gamma_i)$ be the *oldest vertex of* $\Gamma_i$, that is, the vertex that appears in $\Gamma_i, \Gamma_{i-1}, \ldots, \Gamma_{i-j}$ with highest $j$. The decision of switching between GC and GCC (or vice versa) is taken each time you start to draw a new foot of $old(\Gamma_i)$. We begin with Algorithm GC and use points in the clockwise direction with respect to $old(\Gamma_i)$ until we have used enough of them to ensure that the points near to $old(\Gamma_i)$ in the counter-clockwise direction are available. At this point, we switch to Algorithm GCC, starting from the point immediately next to $old(\Gamma_i)$ in the counter-clockwise direction, and we use Algorithm GCC to draw the next feet of $old(\Gamma_i)$ until the last drawn foot of $old(\Gamma_i)$ has used enough points in the counter-clockwise direction to ensure that the points in clockwise direction are available. Figure 2(a) shows an example where three feet were drawn by GC and the fourth foot is drawn by GCC.

Formally, Algorithm SNOWPLOW (SP) works as follows. Let $old_i$ be the index of the point of $\Gamma_i$ where $old(\Gamma_i)$ is drawn. Suppose that edge $e_i = (u, v)$ has to be added to the drawing. If $v$ is present in $\Gamma_{i-1}$ then $\Gamma_i = \Gamma_{i-1}$. Otherwise, if $u \neq old_i$ or $u = old_i$ but $(next_i - old_i) \mod m \leq \lceil \frac{k}{2} \rceil$, place $v$ on $p_{next_i}$ and set $next_{i+1} = (next_i + 1) \mod m$. If $u = old_i$ and $(next_i - old_i) \mod m > \lceil \frac{k}{2} \rceil$, then switch to GCC, that is, place $v$ on point $p_{(old_i - 1) \mod m}$ and set $next_{i+1} = (old_i - 2) \mod m$.

A critical step is when $old(\Gamma_i) \neq old(\Gamma_{i-1})$. This happens when an XL-foot is drawn either by GC or by GCC. In this case the heel of such a foot becomes the oldest vertex (see Fig. 2(b) for an example).

We show in the following that SP needs $2k - 1$ points in convex position to produce a non-crossing drawing of the stream of edges independently of the degree of the vertices.

**Theorem 2.** *Let $S$ be a stream of edges produced by an Eulerian tour of a tree. Algorithm* SP *draws $S$ with persistence $k$ without crossings on $2k - 1$ points in convex position. Also $R_{SP} = O(k^2)$.*

*Sketch of Proof:* Suppose that Algorithm SP is in its GC phase. Assume, without loss of generality, that $p_{old_i} = p_0$, and denote by $P^+ = \{p_1, p_2, \dots p_{\lceil \frac{k}{2} \rceil - 1}\}$ ($P^- = \{p_{-1}, p_{-2}, \dots p_{-\lceil \frac{k}{2} \rceil + 1}\}$) the points after $p_{old_i}$ in clockwise (counter-clockwise) order. Consider the case when $p_{old_i}$ has a sequence of R-feet. In order to switch to the GCC phase at least $\lceil \frac{k}{2} \rceil$ points and at most $2\lceil \frac{k}{2} \rceil - 1$ points of $P^+$ are used. Since at least $\lceil \frac{k}{2} \rceil$ points are used of $P^+$, at least the same amount of time elapsed from when the current GC phase started. Hence, points in $P^-$ are not used by any vertex. □

## 4 Infinite Persistence Drawings of Trees

We consider different scenarios depending on the ordering of the edges in the stream: $(i)$ The edges come in an arbitrary order, with the only constraint that the connectivity is preserved, $(ii)$ the edges come according to a DFS/BFS traversal, $(iii)$ the edges come layer by layer. For each scenario different classes of trees are analyzed.

### 4.1 Arbitrary Order Scenario

In the arbitrary order scenario we first analyze the case of binary trees, then we give results for bounded degree trees and, eventually, for general trees. The following lemma deals with a very simple drawing strategy.

**Lemma 4.** *Let $S = (e_1, e_2, \dots)$ be any stream of edges such that, at each time $i \geq 1$, $G_i$ is a rooted binary tree. Suppose that the root of all $G_i$ is one of the two end-vertices of $e_1$. There exists a drawing algorithm $A$ for $S$ in the infinite persistence model, such that the drawing of any $G_i$ is downward with respect to the root and $R_A = \Theta(2^n)$.*

*Sketch of Proof:* Place the root at $(0, 0)$ and place its first child $v_1$ on $(0, 1)$ and its second child $v_2$ on $(1, 1)$. For every vertex $v$ placed at $(x, y)$ reserve in each subsequent row $z > y$ the points from $x_l = 2^{z-y} \cdot x$ to $x_r = 2^{z-y}(x + 1) - 1$ (see Fig. 3(a)). It is easy to see that an area of $O(n) \times O(2^n)$ is always sufficient for any stream of edges representing a binary tree. The bound is tight since the stream describing the path-like tree of Fig. 3(a) uses an area of $\Omega(n) \times \Omega(2^n)$. Since the best offline algorithm can draw a binary tree in linear area, the statement follows. □

The algorithm in the proof of Lemma 4 is such that whatever is the order in which the edges of a complete binary tree are given, it always computes the same drawing, up to a permutation of the vertex labels. We call such an algorithm a *predefined-location algorithm* for binary trees. Since the competitive ratio of the very simple algorithm in Lemma 4 is exponential, one can ask if there exists a better algorithm that uses a similar strategy. Unfortunately, the next result shows that this is not the case.

**Lemma 5.** *Let $A$ be any predefined-location algorithm for binary trees in the infinite persistence model. Then $R_A = \Omega(2^{\frac{n}{8}}/n)$.*

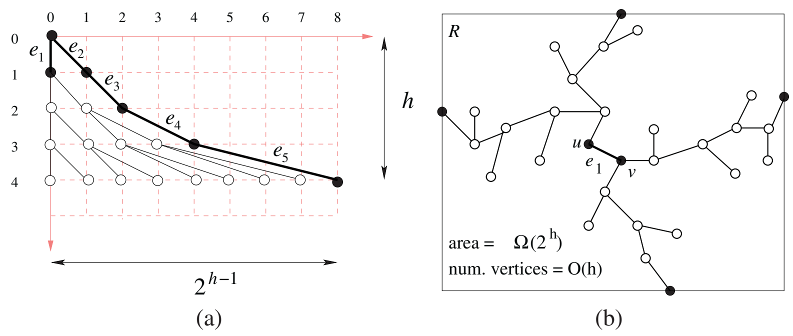

**Fig. 3.** (a) A drawing produced by the algorithm in the proof of Lemma 4. Bold edges represent an edge sequence that causes exponential area. (b) Schematic illustration of the proof of Lemma 5.

*Proof.* We show that there exists a sequence of edges such that the drawing computed by $A$ for the binary tree induced by this sequence requires $\Omega(2^{\frac{n}{8}})$ area, where $n$ is the number of vertices of the tree. Since there exists an offline algorithm that computes a drawing of optimal area $\Theta(n)$ for the binary tree, this implies the statement. Refer to Fig. 3(b). By hypothesis the algorithm always computes the same drawing for a rooted complete binary tree of depth $h$. Consider the bounding box $R$ of such a drawing. Clearly, the area of $R$ is $\Omega(2^h)$. Every path between two vertices of a complete binary tree of depth $h$ consists of at most $2h+1$ vertices and $2h$ edges (the first level has number 0). Independently of the position of the first edge $e_1 = (u, v)$ of the stream, we can define a subsequence of the stream with at most $8h$ edges that forces the algorithm to draw two paths, one consisting of at most $4h$ edges and going from the left side to the right side of $R$, and the other consisting of at most $4h$ edges and going from the bottom side to the top side of $R$, as shown in the figure. Therefore, for this subsequence of $n = 8h$ edges and vertices the algorithm constructs a drawing of area $\Omega(2^h)$. □

If the stream of edges induces at each time a tree whose vertices have degree bounded by a constant $d$, then we can define a drawing algorithm similar to the one described in the proof of Lemma 4. Namely, when a new edge $e = (u, v)$ is processed and $v$ is the $k$-th child of $u$, we set $y(v) = y(u) + 1$ and $x(v) = (d-1) \cdot x(u) + k - 1$. Hence, using the same worst case analysis performed in the proof of Lemma 4, the drawing area used by this algorithm is $\Theta(n) \times \Theta((d-1)^n)$. Since there exists an offline drawing algorithm that takes $\Theta(n)$ area for bounded degree trees [10], we get the following result.

**Lemma 6.** *Let $S = (e_1, e_2, \dots)$ be any stream of edges such that, at each time $i \geq 1$, $G_i$ is a tree with vertex degree at most d. There exists a drawing algorithm A for S in the infinite persistence model, such that $R_A = \Theta((d-1)^n)$.*

The next result extends Lemma 6 to general trees. It proves that there exists an algorithm to draw any infinite tree in the infinite persistence model, under the hypothesis that the stream is connected. In this case we give only a lower bound of the area.

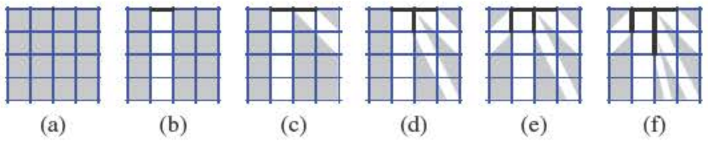
**Fig. 4.** Illustration of the algorithm described in the proof of Lemma 7

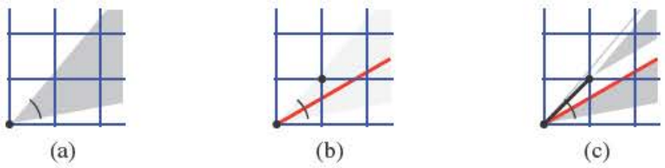
**Fig. 5.** Slicing the cone of a vertex and inserting its child: (a) initial configuration; (b) slicing the cone and finding the closest grid point; (c) inserting the edge.

**Lemma 7.** *There exists an algorithm $A$ that draws in the infinite persistence model any stream of edges $S = (e_1, e_2, \dots)$ such that $G_i$ is a tree of arbitrary vertex degree. The exponential competitive ratio of $A$ is $R_A = \Omega(2^n/n)$.*

*Sketch of Proof:* A greedy drawing strategy is the following (see Fig. 4 and Fig. 5). For each vertex $u$ already placed in the drawing, the algorithm reserves an infinite cone centered at $u$ that does not intersect with any other cone. Each time a new edge $e = (u, v)$ is added to the drawing, the algorithm splits the cone of $u$ into two halves, one of which will be the new cone of $u$ and the other will be used to place $v$ at the first available grid point inside it and to reserve a new (sub-)cone for $v$. Since all the cones assigned to vertices are infinite, it is always possible to add further edges.

The lower bound of the competitive ratio is obtained when using as input the family $G_n = (V_n, E_n)$ defined by $V_n = \{1, \dots, n\}$ and $E_n = \{(i, i+1) \colon 1 \leq i \leq n-2\} \cup \{(n-2, n)\}$. □

## 4.2 BFS and DFS Order Scenarios

If the edges in the stream are given according to some specific order, algorithms can be designed that improve the competitive ratio obtained in the case of the arbitrary order. We focus on orderings deriving from BFS or DFS traversals.

**Lemma 8.** *Let $S = (e_1, e_2, \dots)$ be a stream of edges such that $G_i$ is a tree of any vertex degree, at each time $i \geq 1$, and the edges of the stream are given according to a BFS or to a DFS visit of the graph. There exists a drawing algorithm $A$ for $S$ in the infinite persistence model, such that $R_A = \Theta(n)$.*

*Sketch of Proof:* In the case of the BFS order, we place the first vertex at $(0, 0)$ and all of its $k$ children consecutively along the next row, starting at $(0, 1)$. Processing the children of any vertex at $(x, y)$ we place all its children on the leftmost position that is

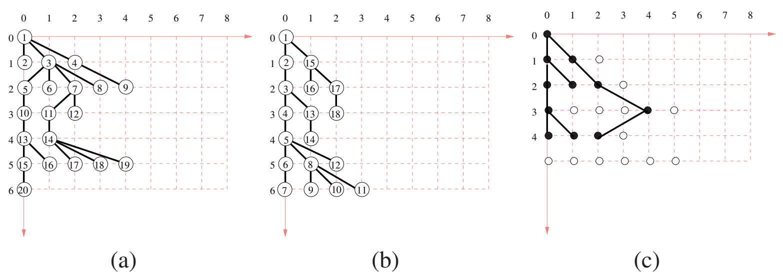

**Fig. 6.** Running examples of drawing algorithms for (a) BFS (b) DFS, and (c) layer ordering

not yet occupied, starting at $(0, y+1)$ (see Fig. 6(a)). The required area is clearly in $O(n^2)$ for this algorithm. For offline algorithms the required area is bound from below by $\Omega(n)$ and thus the statement follows. It can be seen that the worst–case for the BFS order requires a quadratic area, which implies that the analysis is tight. When drawing a tree that comes in DFS order, every vertex can be placed at the leftmost unoccupied position below its parent. The area is $O(n^2)$ and the analysis of the worst case implies that this bound is tight. □

## 4.3 Layer Order

This scenario is intermediate between the BFS order scenario and the arbitrary order one. In the *layer order* scenario edges come layer by layer, but the order of the edges in each layer is arbitrary. We prove the following.

**Lemma 9.** *Let $d > 0$ be a given integer constant and let $S = (e_1, e_2, \dots)$ be any stream of edges such that $G_i$ is a tree of vertex degree at most d, at each time $i \geq 1$, and the edges of the stream are given according to a layer order. There exists a drawing algorithm A for S in the infinite persistence model, such that $R_A = \Theta(dn)$.*

*Proof.* Algorithm $A$ works as follows. If $e_1 = (u, v)$ is the first edge of the stream, set $x(u) = 0$, $y(u) = 0$, $x(v) = 0$, $y(v) = 1$. Also, since $u$ has at most other $d-1$ adjacent vertices, reserve $(d-1)$ consecutive grid points to the right of $v$. When the first vertex of a new level $l$ ($l \geq 1$) enters the drawing, all vertices of the previous level $l-1$ have been already drawn. Hence, if $n_{l-1}$ is the number of vertices of level $l-1$, reserve $(d-1)n_{l-1}$ consecutive grid points for the vertices of level $l$. Namely, denote by $u_1, u_2, \dots, u_{n_{l-1}}$ the vertices at level $l-1$, from left to right. Use the leftmost $(d-1)$ grid points at level $l$ for arranging the children of $u_1$, the next $(d-1)$ grid points for arranging the children of $u_2$, and so on. See Fig. 6(c) for an example. The width of the drawing increases at most linearly with the number of vertices of the tree. Indeed, if the width of the drawing is $w$, there is at least one level $l$ of the drawing having a vertex with $x$-coordinate equal to $w$. This implies that level $l-1$ contains $w/(d-1)$ vertices. Also, since each level contains at least one vertex and since the height of the drawing is equal to the number of levels, the height of the drawing increases at most linearly with the number of vertices. Hence, the area is $O(dn) \times O(n)$. Also, it is easy to find an instance requiring such an area. The best offline algorithm takes $\Theta(n)$ area. □

## 5 Open Problems

This paper opens many possible research directions, including the following: (i) Some of our algorithms have high competitive ratio, hence it is natural to investigate better solutions. (ii) Computing tighter lower bounds would allow us to have a more precise evaluation of streaming algorithms. (iii) It would be interesting to extend the study to larger classes of planar graphs or even to general graphs. (iv) Other persistence models can be considered. For example we could have drawings where the persistence is $O(\log n)$, where $n$ is the size of the stream.

## Acknowledgments

We thank Ioannis G. Tollis for interesting conversations.

## References

1. Bar-Yossef, Z., Kumar, R., Sivakumar, D.: Reductions in streaming algorithms, with an application to counting triangles in graphs. In: Proc. SODA, pp. 623–632 (2002)
2. Bárány, I., Tokushige, N.: The minimum area of convex lattice *n*-gons. Combinatorica 24(2), 171–185 (2004)
3. Biedl, T., Kant, G.: A better heuristic for orthogonal graph drawings. Computational Geometry 9, 159–180 (1998)
4. Branke, J.: Dynamic graph drawing. In: Kaufmann, M., Wagner, D. (eds.) Drawing Graphs. LNCS, vol. 2025, pp. 228–246. Springer, Heidelberg (2001)
5. Buriol, L., Donato, D., Leonardi, S., Matzner, T.: Using data stream algorithms for computing properties of large graphs. In: Proc. Workshop on Massive Geometric Datasets (MASSIVE 2005), pp. 9–14 (2005)
6. Crescenzi, P., Di Battista, G., Piperno, A.: A note on optimal area algorithms for upward drawings of binary trees. Comput. Geom. Theory Appl. 2, 187–200 (1992)
7. de Fraysseix, H., Pach, J., Pollack, R.: How to draw a planar graph on a grid. Combinatorica 10, 41–51 (1990)
8. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 531–543. Springer, Heidelberg (2004)
9. Garg, A., Rusu, A.: Straight-line drawings of binary trees with linear area and arbitrary aspect ratio. In: Goodrich, M.T., Kobourov, S.G. (eds.) GD 2002. LNCS, vol. 2528, pp. 320–331. Springer, Heidelberg (2002)
10. Garg, A., Rusu, A.: Straight-line drawings of general trees with linear area and arbitrary aspect ratio. In: Kumar, V., Gavrilova, M.L., Tan, C.J.K., L'Ecuyer, P. (eds.) ICCSA 2003. LNCS, vol. 2669, pp. 876–885. Springer, Heidelberg (2003)
11. Huang, M.L., Eades, P., Wang, J.: On-line animated visualization of huge graphs using a modified spring algorithm. J. Vis. Lang. Comput. 9(6), 623–645 (1998)
12. Muthukrishnan, S.: Data streams: Algorithms and applications. Foundations and Trends in Theoretical Computer Science 1(2), 117–236 (2005)
13. Papakostas, A., Tollis, I.G.: Interactive orthogonal graph drawing. IEEE Trans. Computers 47(11), 1297–1309 (1998)
14. Shiloach, Y.: Arrangements of Planar Graphs on the Planar Lattice. Ph.D. thesis, Weizmann Institute of Science (1976)