Vol.2020-AL-177 No.4
2020/3/16

Better Embedding of k-Outerplanar Graphs into Random Trees

Akira Matsubayashi[1,a)]

Abstract: Low-distortion embedding of an edge-weighted graph into random trees is addressed. We prove that any k-outerplanar graph can be embedded into random trees with distortion 149^k, improving the previous result with distortion 200^k. The present result implies an embedding of a k-outerplanar graph into ℓ_1-metric with distortion 149^k and better approximation algorithms and online algorithms for some problems on k-outerplanar graphs.

Keywords: low-distortion embedding, probabilistic embedding, k-outerplanar graph, tree metric, Halin graph

1. Introduction

This report addresses the problem of embedding an edge-weighted guest graph into random host trees. More specifically, we are given a graph $G = (V_G, E_G)$ with non-negative edge-weights $w : E_G \to \mathbb{R}_+$ and asked to randomly find a tree $T = (V_T, E_T)$ such that $V_G \subseteq V_T$, and for each pair of vertices u, v in G, the distance (i.e, the length of the shortest path) $d_G(u, v)$ between u and v in G is no shorter than the distance $d_T(u, v)$ in T. The objective of this problem is to minimize the distortion $\rho = \max_{u,v \in V_G} \mathbb{E}[d_T(u, v)]/d_G(u, v)$.

Motivation

Studying this problem is motivated by the fact that any embedding into random trees implies an embedding into ℓ_1-metric. Specifically, if a graph G can be isometrically (i.e., with distortion 1) into random trees, then there exists an isometric ℓ_1-embedding, i.e., a mapping f from the vertex set of G into a real space of some dimension equipped with ℓ_1-norm, such that for any vertices u, v in G, $d_G(u, v) = \|f(u) - f(v)\|_1$ (see, e.g., [8]). One important application of ℓ_1-embeddings is that an ℓ_1-embedding of a graph G with distortion ρ implies an upper bound of the gap between the max-flow and the min-cut for multicommodity flow in G, and hence a ρ-approximation algorithm for the sparsest cut problem [15]. As a stronger result, it is known that the minimum distortion of an ℓ_1-embedding of G and the gap between the max-flow and the min-cut in G are equivalent [11].

Another motivation of embedding into random trees is to provide an approximation (with respect to distance) of a general graph with complicated topology by a simple topology of trees. Based on the fact that many optimization problems and online problems defined on edge-weighted graphs are easily and efficiently solved on trees, low-distortion embeddings into random trees are useful to design randomized approximation algorithms and randomized online algorithms for such problems. Specifically, if there exists an c-approximation (c-competitive, resp.) algorithm on trees for some optimization (online, resp.) problem, and if a graph G can be embedded into random trees with distortion ρ, then we can obtain a $c\rho$-approximation ($c\rho$-competitive, resp.) randomized algorithm on G by the following steps: (i) choose a random tree T into which G is embedded, (ii) run the c-approximation (c-competitive, resp.) algorithm to obtain a solution on T, and (iii) translate the solution obtained on T back into a solution on G [2]. Bartal presented approximation and online algorithms based on this idea for several problems, such as the group Steiner tree problem and the metrical task systems [2], [3]. Carikar et al. derandomized some approximation algorithms by presenting an embedding into a small number of random trees [6].

It should be noted that embeddings of [2], [3], [6] require trees with virtual vertices in addition to the vertices of a guest graph. Thus, the solution obtained in step (ii) above needs to be represented without using virtual vertices. Namely, if some virtual vertices (or anything that must be represented using virtual vertices) are a feasible solution of the problem, then we might not be able to translate the solution back into the guest graph in step (iii). One example of such problems is the metrical task systems. (For this problem, however, an algorithm based on the embedding of [1] is presented [4].)

Previous Results

The currently best embedding of a general graph with n vertices into random trees (with virtual vertices) achieves $O(\log n)$ distortion, which was presented by Fakcharoenphol et al. [10]. This result matches the $\Omega(\log n)$ lower bound for series-parallel graphs [11]. Gupta et al. conjectured that any graph in a class $\mathcal{F}$ is ℓ_1-embeddable with distortion of a constant dependent only on $\mathcal{F}$ if and only if $\mathcal{F}$ forbids a fixed minor [11]. Along the line of this conjecture, embedding for such graph classes are studied via embedding into random trees. It is known that any outerplanar graph can be embedded into random trees with distortion 8

[1] Division of Electrical Engineering and Computer Science, Kanazawa University Kakuma-machi, Kanazawa, 920–1192 Japan
[a)] mbayashi@t.kanazawa-u.ac.jp

[11] (whereas such graphs are isometrically ℓ_1-embeddable [17]). Chekuri et al. extended this result to k-outerplanar graphs by presenting an embedding such a graph into random trees with distortion 200^k [7]. Emek also considered k-outerplanar graphs and presented an embedding into random spanning subtrees with distortion c^k, where c is not explicitly specified but appears to be 244. Lee and Sidiropoulos proves that any graph with pathwidth k can be embedded into random trees with distortion $(4k)^{k^3+1}$ [14].

Contribution and Technique

In this report, we prove that any k-outerplanar graph can be embedded into random trees with distortion 149^k. This improves the result of Chekuri et al. [7] exponentially with respect to k. Our proof is constructive, i.e., we present a randomized embedding algorithm. The proposed algorithm, as well as the algorithms of [7], [9], [11], needs no virtual vertices in host trees, i.e., embeds into random trees with the vertex set of the guest graph. Therefore, the present work can be combined with any approximation or online algorithms designed for trees. For example, we can obtain a randomized $149^k(2 + 1/D)$-competitive algorithm for the file allocation problem [5] on k-outerplanar graphs, by combining with the $(2+1/D)$-competitive algorithm on trees [16], where D is a positive integer representing the size of files managed in the problem. Note that the algorithm of [16] generally allocates files at virtual vertices (if any).

The proposed embedding algorithm is recursive with respect to k, as well as the previous algorithms [7], [9]. Namely, we first embed $(k-1)$-outerplanar graphs obtained from a given k-outerplanar graph by removing the outer cycle into random trees. We then embed the resulting graph, which is essentially a Halin graph, into random trees. The advantage of the proposed algorithm is based on the improvement of the embedding of the Halin graph. The algorithm of [7] first embeds a Halin graph into an outerplanar graph with drastically modifying the topology, and then into random trees. In contrast, the proposed algorithm adds new edges, essentially, in a quite limited situation. In this sense, the generated random trees are different from but close to spanning subtrees, and more intuitive.

2. Preliminaries

Graph Metric and Embedding

Graphs considered in this paper are simple and undirected, and have non-negative edge-weights, $w(e) \geq 0$ for any edge e. For a graph G, we denote its vertex set and edge set by V_G and E_G, respectively. We use the notation of w also for graphs, i.e., $w(G) := \sum_{e \in E_G} w(e)$. Let $d_G(u, v)$ be the distance of vertices u and v in G, i.e., the minimum sum of weights of edges in a path connecting u and v in G. The distance d_G is called the *graph metric supported on* G. An edge-weighted graph H *dominates* G if $V_G = V_H$ and $d_G(u, v) \leq d_H(u, v)$ for any vertices $u, v \in V_G$. The graph G is *embedded into random trees* with *distortion* ρ (or ρ-*probabilistically approximated by trees*) if there is a probability distribution $\mathcal{D}$ over a set of trees T dominating G such that for any vertices $u, v \in V_G$, the expected distance between u and v in T under the distribution $\mathcal{D}$ is at most $\rho \cdot d_G(u, v)$, i.e., the following condition is satisfied.

Condition 1 *For any* $u, v \in V_G$, $\mathbb{E}_{\mathcal{D}}[d_T(u, v)] \leq \rho \cdot d_G(u, v)$.

It should be noted that this condition can be replaced with the following condition.

Condition 2 *For any* $(u, v) \in E_G$, $\mathbb{E}_{\mathcal{D}}[d_T(u, v)] \leq \rho \cdot d_G(u, v)$.

Because any edge (u, v) with $w((u, v)) > d_G(u, v)$ does not affect the graph metric d_G, without loss of generality, we assume that $d_G(u, v) = w((u, v))$ for any edge (u, v). By this assumption, Condition 2 can be replaced with the following condition.

Condition 3 *For any* $(u, v) \in E_G$, $\mathbb{E}_{\mathcal{D}}[d_T(u, v)] \leq \rho \cdot w((u, v))$.

The following simple facts, used in previous work [7], [9], are used in this paper without proofs.

Lemma 1 *If each 2-connected component H of a graph G can be embedded into random trees with distribution ρ, then so can G.*

Lemma 2 *If a subgraph H of a graph G can be embedded into random trees with distortion ρ_1, and if the graph G_T obtained from G by replacing the edges of H with the edges of any tree T dominating H can be embedded into random trees with distortion ρ_2, then G can be embedded into random trees with distortion $\rho_1\rho_2$.*

Graph Classes

A planar graph G is 1-*outerplanar*, or simply *outerplanar*, if it can be drawn in the plane so that all the vertices belong to the unbounded (outer) face. For $k \geq 2$, a planar graph G is k-*outerplanar* if it has a planar drawing such that removing vertices and edges in the outer face results in a $(k-1)$-outerplanar graph. A vertex or an edge of G is said to be *outer* if it is contained in the outer face, *inner* otherwise.

A *Halin* graph was originally introduced as a planar graph obtained from a planar drawing of a tree with at least four vertices and without degree-2 vertices by adding edges joining leaves of the tree into a cycle [12], [13]. In this paper, we adopt a slightly different definition: A Halin graph is defined as a planar graph obtained from a planar drawing of a tree with at least two leaves by adding paths connecting leaves of the tree into a path. It should be noted that the definition of this paper is essentially a generalization (contrary to appearance) of original Halin graphs. Indeed, we can modify any original Halin graph to a graph following the definition of this paper and essentially having the same graph metric, by subdividing each of the two outer edges (u, v) incident to some vertex u into (u, v') of weight $w((u, v))$ and (v', v) of weight 0. We also utilize more generalized graphs defined in [9]: A *bush* is a graph that has a planar drawing such that removing vertices and edges in its outer face results in a forest. By definition, a Halin graph is 1- or 2-outerplanar, and a bush is 2-outerplanar. Examples of a Halin graph and a bush are shown in Fig. 1.

3. Embedding Algorithm

In this section, we describe a randomized algorithm, called `Ek0G`, in a top-down fashion. This algorithm takes a k-outerplanar

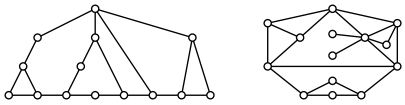

Fig. 1 A Halin graph (left) and a bush (right).

graph G as input, together with a planar drawing, and generates a random tree dominating G. Without loss of generality, we assume that the k-outerplanar graph input to EkOG has no edges of weight 0. An embedding of G into random trees is implied by the probability distribution over the trees generated by the algorithm.

3.1 Main Structure

3.1.1 Onion Peeling

The algorithm EkOG has the same structure as previous work [7], [9] so-called *onion peeling*. That is, for each 2-connected component D of G, we recursively generate random trees dominating $(k-1)$-outerplanar subgraphs of D obtained by removing the outer face from D, replace the $(k-1)$-outerplanar subgraphs with the generated trees, and for the resulting graph B, generate a random tree dominating B. It should be noted that the graph B is a bush. For convenience of description of the embedding algorithm, we assume without loss of generality that B is a Halin graph, together with attached trees and some properties, as described in the following section. Note that this assumption is not essential, i.e., we could design our algorithm for embedding a bush with the same distortion.

3.1.2 Assumption for Bushes

Any bush has a unique 2-connected component C having a cycle. We call such a unique component of a bush a *core*. Without loss of generality, we assume that C has the following properties.

Property 1 *C is a Halin graph, i.e., C consists of a tree T with some root r and a path P connecting the leaves of T.*

Property 2 *At most one of any two adjacent edges in a path in T between r and a leaf has weight* 0*.*

Property 3 *At most one of any two paths in T between r and two leaves has weight* 0*.*

Indeed, we can modify any core C to a graph having these properties by the following procedure, so that the graph metric is preserved, and any embedding of C is not essentially affected. Let R be the cycle in the outer face of C and F be the graph induced by $E_C \setminus E_R$.

(1) Choose one vertex $r \in V_F \cap V_R$ as the root, and subdivide each of two edges (r, v_i) in R ($i \in \{1,2\}$) into (r, v_i') of weight $w((r, v_i))$ and (v_i', v_i) of weight 0. Let T be $F \cup (r, v_1') \cup (r, v_2')$ and P be the path induced by $E_R \setminus \{(r, v_1'), (r, v_2')\}$.

(2) If there is a vertex $v \in V_T \cap V_P$ incident to two edges e_1 and e_2 in P and to $j \geq 2$ edges in T, then we replace v with a path $(v_1, \ldots, v_j)$ of weight 0, so that v_1 and v_j are incident to e_1 and e_2, respectively, and v_i $(1 \leq i \leq j)$ is incident to exactly one of the j edges in F (originally incident to v) with preserving planarity. At this point, any vertex $v \in V_T \cap V_P$ is incident to two edges in P and to one edge in T. This implies that T becomes a forest. Moreover, only edges in P have weight 0.

(3) If there is an edge in T joining two vertices u and v in P, then we add new vertices u' and v' and replace the edge (u, v) with three edges (u, u') of weight 0, (u', v') of weight $w(u, v)$, and (v', v) of weight 0. At this point, there are no edges in T joining two vertices in P, and Properties 2 and 3 hold.

(4) If T consists of two or more disconnected trees, then there exists a face containing edges of some disconnected trees T_1 and T_2 in T. By Steps (2) and (3), this face contains at least one non-leaf vertex v_i of T_i for each $i \in \{1,2\}$ such that $d_C(v_1, v_2)$ is positive. Adding a new edge (v_1, v_2) of weight $d_C(v_1, v_2)$ decrements the number of trees in T by 1 without violating Properties 2 and 3. We continue this process until T consists of exactly one tree, i.e., C becomes a Halin graph.

It should be noted that since the vertices added in Steps (2) and (3) are incident to edges of weight 0, and since any embedding algorithm with finite distortion never removes these edges, we naturally remove the added vertices by contracting these weight-0 edges from a dominating tree of C generated by the embedding algorithm.

With the above assumption, we embed the core C using the algorithm, called EHG and defined in Sect. 3.2, for embedding a Halin graph. The following is a formal description of the main structure of EkOG.

EkOG(G)
Input: A k-outerplanar graph G.
Output: A random tree dominating G.

(1) For each 2-connected component D of a k-outerplanar graph G, perform the following steps.
  (a) If D is 1-outerplanar, generate a random tree T_D dominating D using the algorithm of [11].
  (b) If D is k-outerplanar with $k \geq 2$, then let $D_1, \ldots, D_\ell$ be $(k-1)$-outerplanar graphs obtained by removing vertices in the outer face of D, and perform the following steps.
    (i) For $1 \leq i \leq \ell$, perform EkOG(D_i) recursively to generate a random tree T_i dominating D_i.
    (ii) Let B be the bush obtained from D by replacing the edges in D_i with the edges in T_i for $1 \leq i \leq \ell$.
    (iii) Let C be the core of B and perform EHG(C) (defined in Sect. 3.2) to generate a random tree T_C dominating C.
    (iv) Let T_D be the tree obtained from B by replacing the edges in C with the edges in T_C.

(2) Return the union of the generated trees T_D dominating the 2-connected components D of G.

3.2 Embedding of Halin Graphs

3.2.1 Preliminaries

The following definitions will be used to describe the embedding algorithm for Halin graphs. Let H be a Halin graph to be embedded that consists of a tree T and a path Q connecting the leaves of T. We define a vertex $r \in V_T \setminus V_Q$ as a root of T. For a vertex $p \in V_T$, let T_p be the subtree of T induced by p and all the descendants of p in T. The minimal subpath of Q containing all the leaves of T_p is called the *base of p* and denoted by Q_p. The

union of T_p and Q_p is called the *delta of* p and denoted by Δ_p. Note that $T_r = T$, $Q_r = Q$, and $\Delta_r = H$. Let ℓ_p be a leaf of T_p at the minimum distance from p. The path between p and ℓ_p in T_p is called a *spine of* p and denoted by S_p. We fix ℓ_p and S_p while processing Δ_p in the embedding algorithm. We also fix the root r until the embedding of H is completed. The graph Δ_p is *narrow* if for any vertex u in T_p and any leaf ℓ of T_u, the distance between ℓ_u and ℓ in Q_u is at most α times the distance between ℓ_u and ℓ in T_u, i.e., $d_{Q_u}(\ell_u, \ell) \leq \alpha \cdot d_{T_u}(\ell_u, \ell)$, where $\alpha > 1$ is a parameter determined later. It should be noted that if Δ_p is narrow, then Δ_u is narrow for any descendant u of p. The graph Δ_p is *wide* if the following conditions hold.

- For any child u of p, Δ_u is narrow.
- There is a leaf ℓ of T_p such that $d_{Q_p}(\ell_p, \ell) > \alpha \cdot d_{T_p}(\ell_p, \ell)$.

It should be noted that if Δ_p is not narrow, then there exists a vertex u in T_p such that Δ_u is wide.

3.2.2 Main Structure of Embedding of Halin Graphs

The algorithm for embedding Halin graphs, called `EHG`, reduces an input Halin graph by replacing a wide delta of a vertex p with a random tree dominating the delta. The random tree is chosen so that p is contained in the path connecting the end-vertices of the base of p. Therefore, the resulting graph is a bush whose core is a reduced Halin graph with p in the outer face. Repeating this process, we obtain a Halin graph that is a narrow or wide delta of the fixed root, and finish the embedding by generating a random tree dominating the delta of the root. The algorithms for embedding narrow and wide deltas are defined in Sects. 3.3 and 3.4, respectively.

The following is a formal description of the main structure of `EHG` for an input Halin graph H with a fixed root r.

`EHG`(H)

Input: A Halin graph H with a fixed root r.

Output: A random tree dominating H.

(1) If Δ_r is narrow, then perform `NarrowEmbedding`(Δ_r) (defined in Sect. 3.3) to generate a random tree dominating Δ_r, and return the tree.
(2) Find a vertex p in T_r such that Δ_p is wide, and perform `WideEmbedding`(Δ_p) (defined in Sect. 3.4) to generate a random tree $\tilde{T}_p$ dominating Δ_p in which the path connecting the end-vertices of Q_p contains p.
(3) If $p = r$, then return $\tilde{T}_p$.
(4) Let B be the bush obtained from H by replacing the edges of Δ_p with the edges of $\tilde{T}_p$.
(5) Let C be the unique core of B and perform `EHG`(C) recursively to generate a random tree T_C dominating C.
(6) Let T_H be the tree obtained from B by replacing the edges in C with the edges in T_C, and return T_H.

3.3 Embedding of Narrow Deltas

The algorithm for embedding a narrow delta Δ_p, called `NarrowEmbedding`, keeps the edge joining p and its child u in the spine S_p at probability 1 and generates a random tree $\tilde{T}_u$ by using `NarrowEmbedding` recursively. This implies that $\tilde{T}_u$ has the spine S_p. The tree $\tilde{T}_u$ is chosen so that it also contains the base Q_u. For another child v of p, the algorithm separates $(p, v) \cup \Delta_v$ into random two trees by cutting edges in $(p, v) \cup T_v$, so that p is contained in one of the separated two trees, and Q_u is contained in the other tree. Therefore, the graph generated from Δ_p is a tree dominating Δ_p.

The separating procedure for $(p, v) \cup \Delta_v$, called `NarrowCut`, removes the edge (p, v) at probability

$$1 - \left[\frac{w(S_v)}{w((p, v)) + w(S_v)}\right]^2.$$

If (p, v) was removed, then Δ_v is embedded using `NarrowEmbedding`. Otherwise, Δ_v is separated recursively.

The following two matters should be noted. The first one is that if $w(S_v) = 0$ and $w((p, v)) > 0$, then (p, v) is removed at probability 1, and therefore, no further recursive `NarrowCut` is performed. In other words, if $w((p, v)) > 0$ and (p, v) is not removed, then $w(S_v) > 0$ and v has a child. The second matter is that since we introduce edges of weight 0 joining outer and inner vertices to assume Properties 1–3, it may be the case that $w(S_v) = w((p, v)) = 0$, which makes the probability of removing (p, v) an indeterminate form. In this case, however, `NarrowCut` is not performed for $(p, v) \cup \Delta_v$ by the definition of the algorithm and the procedure to assume Properties 1–3, This is proved as Lemma 3 in Sect. 4.1. Therefore, we may assume that $w(S_v) > 0$ or $w((u, v)) > 0$ in `NarrowCut`.

The following are formal descriptions of `NarrowEmbedding` and `NarrowCut`.

`NarrowEmbedding`(Δ_p)

Input: A narrow delta Δ_p of a vertex p.

Output: A random tree dominating Δ_p and containing the spine S_p and base Q_p.

(1) If T_p is the single vertex p, then return p.
(2) For the child u of p in the spine S_p of p, perform `NarrowEmbedding`(Δ_u) to generate a random tree $\tilde{T}_u$ dominating Δ_u and containing Q_u.
(3) For a child v of p not in S_p, perform `NarrowCut`($(p, v) \cup \Delta_v$) to generate a random forest F_v that dominates $(p, v) \cup \Delta_v$ and consists of two trees containing p in one tree and Q_v in the other tree.
(4) Return the tree obtained from Δ_p by replacing the edges of Δ_u with the edges of $\tilde{T}_u$, and the edges of $(p, v) \cup \Delta_v$ with the edges of F_v for every child v not in S_p.

`NarrowCut`($(p, v) \cup \Delta_v$)

Input: The graph $(p, v) \cup \Delta_v$ of a vertex p, a child v of p, and the delta Δ_v of v.

Output: A random forest that dominates $(p, v) \cup \Delta_v$ and consists of two trees containing v in one tree and the base Q_v in the other tree.

(1) Remove the edge (p, v) at probability $1 - [w(S_v)/(w((p, v)) + w(S_v))]^2$.
(2) If (p, v) was removed, then perform `NarrowEmbedding`(Δ_v) to generate a random tree $\tilde{T}_v$ dominating Δ_v and containing Q_v. Then, return $p \cup \tilde{T}_v$.
(3) If (p, v) remains, then for each child x of v, perform `NarrowCut`($(v, x) \cup \Delta_x$) to generate a random forest F_x that dominates $(v, x) \cup \Delta_x$ and consists of two trees containing v in one tree and Q_x in the other tree. Then, return the tree obtained from $(p, v) \cup \Delta_v$ by replacing the edges of $(v, x) \cup \Delta_x$

with the edges of F_x for each child x of v.

3.4 Embedding of Wide Deltas

The algorithm for embedding a wide delta Δ_p, called `WideEmbedding`, first finds a set U of children of p such that

- for any distinct $u, v \in U$, $d_{Q_p}(\ell_u, \ell_v) > \alpha \cdot d_{T_p}(\ell_u, \ell_v)$;
- for any child $x \notin U$ of p, there is $u \in U$ such that $d_{Q_p}(\ell_u, \ell_x) \leq \alpha \cdot d_{T_p}(\ell_u, \ell_x)$ and $d_{T_p}(p, \ell_u) \leq d_{T_p}(p, \ell_x)$.

The set U can be found by a simple greedy search, which is described specifically in a formal description below. In particular, the child of p in the spine S_p is chosen as an element of U. Second, the algorithm performs `NarrowEmbedding`(Δ_u) for each $u \in U$ and `NarrowCut`$((p, x) \cup \Delta_x)$ for each child $x \notin U$ of p. At this point, we obtain a random graph containing the base Q_p and the paths connecting p and ℓ_u for all children $u \in U$ of p.

Third, if $|U| = 1$, then the algorithm chooses one leaf v with $d_{Q_p}(\ell_p, v) > \alpha \cdot d_{T_p}(\ell_p, v)$ and adds a new edge (p, v) of weight $d_{T_p}(p, v)$ to the random graph obtained thus far. Moreover, v is added to U. The vertex v becomes a child of p in the resulting graph. We define $\ell_v := v$.

At this point, we have at least two vertices in U. Two vertices $u, v \in U$ are said to be *consecutive* if there is no vertex $x \in U$ such that ℓ_x appears between ℓ_u and ℓ_p in Q_p. Finally, for each pair of consecutive $u, v \in U$, the algorithm chooses and removes one edge e in the subpath P of Q_p between ℓ_u and ℓ_v at probability $w(e)/w(P)$. Then, a tree dominating Δ_p is obtained. Moreover, the path in the tree connecting end-vertices of Q_p contains p.

The following is a formal description of `WideEmbedding`.

`WideEmbedding`(Δ_p)

Input: A wide delta Δ_p of a vertex p.

Output: A random tree dominating Δ_p in which the path connecting the end-vertices of the base Q_p contains p.

(1) Find a set U of children of p as follows.
  (a) Let $U := \{v\}$ for the child v of p in the spine S_p, and mark the other children of p "unchecked".
  (b) Remove the mark of each "unchecked" child x of p with $d_{Q_p}(\ell_u, \ell_x) \leq \alpha \cdot d_{T_p}(\ell_u, \ell_x)$.
  (c) Include an "unchecked" child u of p with the minimum $d_{T_p}(p, \ell_u)$ to U, and remove the mark of u.
  (d) Repeat (b) and (c) while "unchecked" children of p remain.

(2) For each $u \in U$, perform `NarrowEmbedding`(Δ_u) to generate a random tree $\tilde{T}_u$ dominating Δ_u and containing Q_u.

(3) For each child $x \notin U$ of p, perform `NarrowCut`$((p, x) \cup \Delta_x)$ to generate a random forest F_x that dominates $(p, x) \cup \Delta_x$ consists of two trees containing p in one tree and Q_x in the other tree.

(4) Let $\tilde{\Delta}_p$ be the graph obtained from Δ_p by replacing the edges of Δ_u with the edges of $\tilde{T}_u$ for each $u \in U$, and the edges of $(p, x) \cup \Delta_x$ with the edges of F_x for each child $x \notin S$ of p.

(5) If $|U| = 1$, then perform the following:
  (a) Choose one leaf v of T_p with $d_{Q_p}(\ell_p, v) > \alpha \cdot d_{T_p}(\ell_p, v)$ arbitrarily.
  (b) Add a new edge (p, v) of weight $d_{T_p}(p, v)$ to $\tilde{\Delta}_p$, and add v to U.
  (c) Define $\ell_v := v$.

(6) For each pair of consecutive $u, v \in U$, choose one edge e in the subpath P of Q_p between ℓ_u and ℓ_v at probability $w(e)/w(P)$, and remove e from $\tilde{\Delta}_p$. Then, return the resulting tree.

4. Analysis of Embedding

4.1 Correctness of Embedding

As mentioned in Sect. 3.3, `NarrowCut`$((p, v) \cup \Delta_v)$ is not well-defined if $w(S_p) = w((p, v)) = 0$. We claim in Lemma 3 below that if $w(S_p) = w((p, v)) = 0$, then the proposed algorithm does not perform `NarrowCut`$((p, v) \cup \Delta_v)$. On condition that the claim hold, the correctness of the proposed algorithm `EkOG` and the subroutines that they generate desired random graphs is self-explanatory and can be proved by simple induction based on their definitions. Therefore, we omit the proof for the correctness.

Lemma 3 *`NarrowCut`$((p, v) \cup \Delta_v)$ is not performed if $w(S_v) = w((p, v)) = 0$.*

Proof Assume $w(S_v) = w((p, v)) = 0$. Then, the spine S_p has weight 0 because $w(S_p) \leq w((p, v)) + w(S_p) = 0$. If `NarrowCut`$((p, v) \cup \Delta_v)$ is performed, then one of the following procedures is performed:

- `NarrowCut`$((q, p) \cup \Delta_p)$ for the parent q of p,
- `NarrowEmbedding`(Δ_p), or
- `WideEmbedding`(Δ_p).

Assume first that `NarrowCut`$((q, p) \cup \Delta_p)$ is performed. Since $w((p, v)) = 0$, the edge (q, p) has a positive weight by Property 2. Since $w(S_p) = 0$ as already claimed, the edge (q, p) is removed at probability $1 - [w(S_p)/(w((q, p)) + w(S_p))]^2 = 1$, and by the definition of the algorithm, `NarrowCut`$((p, v) \cup \Delta_v)$ is not performed.

Assume next that `NarrowEmbedding` or `WideEmbedding` is performed for Δ_p. Since $w(S_p) = 0$ and $w((p, v)) + w(S_v) = 0$, v is in S_p by Property 3. Therefore, by the definition of the algorithm, not `NarrowCut`$((p, v) \cup \Delta_v)$ but `NarrowEmbedding`(Δ_v) with preserving the edge (p, v) is performed. (It should be noted that the child v of p in the spine S_p is chosen as an element of U in Step 1a of `WideEmbedding`.) □

4.2 Distortion

For an embedding of a graph G with distribution $\mathcal{D}$ over a set of trees T dominating G, and for an edge (u, v) in G, we call the value $\mathbb{E}_{\mathcal{D}}[d_T(u, v)]/w((u, v))$ the *distortion of* (u, v) in the embedding. It should be noted that the distortion of the embedding is the maximum distortion of any edge in G.

Suppose that a Halin graph H with a root r is input to `EHG`. Let p be a vertex of T_r such that Δ_p is narrow and input to `NarrowEmbedding` (i.e., $p = r$ in this case), or wide and input to `WideEmbedding`. We divide the edges of Δ_p into three groups as follows. If Δ_p is narrow, then we define E^U to be the set of edges in the spine S_p. If Δ_p is wide, then we define E^U to be the set of edges in the paths in T_p connecting p and ℓ_u for all $u \in U$. If a new edge joining p and a leaf v is added in Step 5b of `WideEmbedding`, then the edge (p, v) is also added to E^U. Let E^T be the set of edges in T_p but not in E^U. Then, the edge sets

E_{Q_p}, E^U, and E^T are distinct, and E_{Δ_p}, possibly together with the additional edge (p, v), is equal to $E_{Q_p} \cup E^U \cup E^T$. Recall that after `NarrowEmbedding` or `WideEmbedding` for Δ_p is finished, unless $p = r$, we obtain a bush B dominating H and perform `EHG`(C) recursively for the unique core C of B.

Let $\delta := 2\alpha/(\alpha - 1) = 2 + 2/(\alpha - 1)$.

Lemma 4 *The distortion of an edge in the base Q_p is at most δ.*

Proof Suppose that $h \geq 0$ recursive calls of `EHG` are performed in total to finish `EHG`(H). We prove the lemma by induction on h.

We first assume $h = 0$, i.e., $p = r$. If Δ_p is narrow, then any edge $e \in E_{Q_p}$ is an edge of the spine of p. Because `NarrowEmbedding`(Δ_p) removes no edge of the spine S_p, the distortion of e is 1. If Δ_p is wide, then any edge $e \in E_{Q_p}$ is possibly removed in Step 6 of `WideEmbedding`(Δ_p), i.e., only if there are consecutive $u, v \in U$ such that e is contained in the subpath P of Q_p between ℓ_u and ℓ_p. Because the probability of the removal is $w(e)/w(P)$, the distortion of e is at most

$$\begin{aligned}
&\left[\left\{d_{Q_p}(\ell_u, \ell_v) + d_{T_p}(\ell_u, \ell_v)\right\}\frac{w(e)}{w(P)} + w(e)\right]/w(e)\\
<&\left\{d_{Q_p}(\ell_u, \ell_v) + \frac{d_{Q_p}(\ell_u, \ell_v)}{\alpha}\right\}\frac{1}{d_{Q_p}(\ell_u, \ell_v)} + 1\\
=&2 + \frac{1}{\alpha} < 2 + \frac{2}{\alpha - 1} = \delta.
\end{aligned}$$

We next assume $h \geq 1$ and the lemma for $h - 1$. Then, Δ_p is wide, and e is either between ℓ_u and ℓ_v for some consecutive $u, v \in U$ or between an end-vertex of Q_p and ℓ_x ($x \in U$) nearest to the end-vertex.

In the former case, e is possibly removed similarly to the case of $h = 0$. The path in T_p from p to ℓ_u or to ℓ_v remains in the bush obtained in Step 4 of `EHG`(H), and possibly becomes a part of the base of the core C in the subsequent step. Therefore, the path connecting ℓ_u and ℓ_v in T_p has an expected weight at most $\delta \cdot d_{T_p}(\ell_u, \ell_v)$ by induction hypothesis. It should be noted that if e is not removed, then it is not contained in C, i.e., there is no further chance for e to be removed. Thus, the distortion of e is at most

$$\begin{aligned}
&\left[\left\{d_{Q_p}(\ell_u, \ell_v) + \delta \cdot d_{T_p}(\ell_u, \ell_v)\right\}\frac{w(e)}{d_{Q_p}(\ell_u, \ell_v)} + w(e)\right]/w(e)\\
<&\left\{d_{Q_p}(\ell_u, \ell_v) + \delta \cdot \frac{d_{Q_p}(\ell_u, \ell_v)}{\alpha}\right\}\frac{1}{d_{Q_p}(\ell_u, \ell_v)} + 1\\
=&2 + \frac{\delta}{\alpha} = 2 + \frac{2\alpha}{\alpha - 1}\cdot\frac{1}{\alpha} = \delta.
\end{aligned}$$

In the latter case, i.e., if e is between an end-vertex of Q_p and ℓ_x ($x \in U$) nearest to the end-vertex, then e is contained in the path connecting the end-vertices of Q_p in the core C and becomes an edge of the base of C. Therefore, e has distortion at most δ by induction hypothesis. □

Lemma 5 *The distortion of an edge in E^U is at most δ.*

Proof If Δ_p is narrow, then any edge $e \in E^U$ is an edge of the spine of p. Because `NarrowEmbedding`(Δ_p) removes no edge of the spine S_p, the distortion of e is 1. If Δ_p is wide, then any edge $e \in E^U$ is either an edge joining p and its child $u \in U$ or an edge of a spine S_u of some $u \in U$. In the former case, `WideEmbedding`(Δ_p) does not remove e. In the latter case, e is not removed either because `NarrowEmbedding`(Δ_u) removes no edge of the spine S_u. Thus, the edge e remains in the bush obtained in Step 4 of `EHG`, and either becomes an edge of the core C in the subsequent step or is not contained in C. In the former case, e is an edge of the base of C and has distortion at most δ by the Lemma 4. In the latter case, the distortion of e is obviously 1. □

To estimate the distortion of edges in E^T, we fix $e \in E^T$ and suppose that e joins a vertex v_0 and its parent v_1. We examine properties, expected weights, and probabilities of random paths in T_p connecting v_0 and v_1 for the case of removal of e.

Let $(v_1, \ldots, v_h)$ be the maximal path in T_p consisting of ancestors of v_0 and edges in E^T. It should be noted that $v_h = p$, or the path connecting v_h and p in T_p consists of edges in E^U. If the edge $e = (v_1, v_0)$ is removed in `NarrowEmbedding`(Δ_p) or `WideEmbedding`(Δ_p), then `NarrowCut`($(v_1, v_0) \cup \Delta_{v_0}$) is performed as a descendant procedure. This implies one of the following cases by the definitions of procedures performing `NarrowCut`.

Case 1 `NarrowEmbedding`(Δ_{v_1}) is performed. In this case, `NarrowEmbedding`(Δ_u) is also performed for the child $u \neq v_0$ of v_1 in the spine S_{v_1}.

Case 2 `NarrowCut`($(v_2, v_1) \cup \Delta_{v_1}$) is performed and does not remove the edge (v_2, v_1).

Case 3 Δ_p is wide, $v_1 = p$ (i.e., $h = 1$), and $v_0 \notin U$ in `WideEmbedding`(Δ_p). In this case, `NarrowEmbedding`(Δ_u) is performed for a child $u \in U$ of p such that $d_{T_p}(p, \ell_u) \leq d_{T_p}(p, \ell_{v_0})$ and $d_{Q_p}(\ell_u, \ell_{v_0}) \leq \alpha \cdot d_{T_p}(\ell_u, \ell_{v_0})$.

Lemma 6 *There exists $1 \leq m \leq h$ such that the following condition is satisfied.*

Condition 4
- *For each $2 \leq i \leq m$, `NarrowCut`($(v_i, v_{i-1}) \cup \Delta_{v_{i-1}}$) is performed and does not remove the edge (v_i, v_{i-1}).*
- *There exists a child $u_m \neq v_{m-1}$ of v_m such that*

$$d_{T_p}(v_m, \ell_{u_m}) \leq d_{T_p}(v_m, \ell_{v_{m-1}}), \tag{1}$$

$$d_{Q_p}(\ell_{u_m}, \ell_{v_{m-1}}) \leq \alpha \cdot d_{T_p}(\ell_{u_m}, \ell_{v_{m-1}}), \tag{2}$$

  the edge (v_m, u_m) is not removed, and `NarrowEmbedding`(Δ_{u_m}) is performed.
- *The vertex v_{m-1} is not in the spine S_{v_m}.*

Proof If Case 1 or 3 holds, then Condition 4 is satisfied for $m = 1$. It should be noted that v_0 is not in the spine S_{v_1} in both Cases 1 and 3. If Case 2 holds, then one of Cases 1–3 holds for $v_0, \ldots, v_{h-1}$ with shifted suffixes as $v_i := v_{i+1}$. Repeating this argument, Condition 4 is satisfied for some m.

□

It should be noted that m is a random variable, whose value is determined by random choices performed in `NarrowEmbedding`(Δ_p) or `WideEmbedding`(Δ_p). Moreover, m is unique. This is because if `NarrowCut`($(v_{m+1}, v_m) \cup \Delta_{v_m}$)

is performed and does not remove the edge (v_{m+1}, v_m), then `NarrowCut`$((v_m, x) \cup \Delta_x)$ is performed for every child x of v_m, which implies that either the edge (v_m, u_m) is removed, or `NarrowEmbedding`(Δ_{u_m}) is not performed.

If the edge $e = (v_1, v_0)$ is removed by `NarrowCut`$((v_1, v_0) \cup \Delta_{v_0})$, then `NarrowEmbedding`$(\Delta_{v_0})$ is performed. `NarrowEmbedding` performed for a delta of a vertex x removes no edge of the spine of x. Therefore, at the point that `NarrowEmbedding`(Δ_p) or `WideEmbedding`(Δ_p) is finished, the path connecting v_0 and v_1 for the case of removal of e, called the *alternative path* A_m, is the concatenation of

(i) the edges (v_1, v_2), (v_2, v_3), …, and (v_{m-1}, v_m),
(ii) the edge (v_m, u_m) and the spine S_{u_m},
(iii) the subpath P_m of the base Q_p between ℓ_{u_m} and ℓ_{v_0}, which are the end-vertices of S_{u_m} and S_{v_0} in Q_p, respectively, and
(iv) the spine S_{v_0}.

Lemma 7 *At the point that `EHG`(H) is finished, the expected weight of A_m is $\mathbb{E}[w(A_m)] \le \{1 + (1 + 4\alpha)\delta\} w(D_m)$.*

Proof Because the edges in (i) and (iv) are in E^T, there is no further chance for the edges to be removed. Because the edges in (ii) are contained in $E^U \cup E^T$, they have distortion at most δ by Lemma 5. The edges in (iii) have distortion at most δ by Lemma 4. Thus, the expected weight of A_m is at most

$$\mathbb{E}[w(A_m)] \le w(D_m) + \{w((v_m, u_m)) + w(S_{u_m}) + w(P_m)\}\,\delta, \quad (3)$$

where for $1 \le i \le m$, D_i is the path consisting of the spine S_{v_0} and edges $(v_0, v_1), \ldots, (v_{i-1}, v_i)$. By (1) and $w(S_{v_{m-1}}) \le w(D_{m-1})$,

$$\begin{aligned} w((v_m, u_m)) + w(S_{u_m}) &\le w((v_m, v_{m-1})) + w(S_{v_{m-1}}) \\ &\le w((v_m, v_{m-1})) + w(D_{m-1}) = w(D_m). \end{aligned} \quad (4)$$

Similarly,

$$\begin{aligned} w(P_m) &= d_{Q_p}(\ell_{u_m}, \ell_{v_0}) \le d_{Q_p}(\ell_{u_m}, \ell_{v_{m-1}}) + d_{Q_p}(\ell_{v_{m-1}}, \ell_{v_0}) \\ &\le \alpha \cdot d_{T_p}(\ell_{u_m}, \ell_{v_{m-1}}) + \alpha \cdot d_{T_p}(\ell_{v_{m-1}}, \ell_{v_0}) \\ &\qquad \text{[by (2), and because } \Delta_{v_{m-1}} \text{ is narrow]} \\ &\le \alpha\{w((v_m, u_m)) + w(S_{u_m}) + w((v_m, v_{m-1})) + w(S_{v_{m-1}})\} \\ &\quad + \alpha\{w(S_{v_{m-1}}) + w(D_{m-1})\} \\ &\le \alpha\{2w(D_m) + 2w(D_{m-1})\} \quad \text{[similarly to (4)]} \\ &\le 4\alpha \cdot w(D_m) \qquad \text{[by } w(D_{m-1}) \le w(D_m)\text{]}. \end{aligned} \quad (5)$$

Combining (3)–(5), we have $\mathbb{E}[w(A_m)] \le \{1 + (1 + 4\alpha)\delta\} w(D_m)$. □

Let M be the set of numbers $1 \le i \le h$ such that v_{i-1} is not contained in the spine of v_i. It should be noted that $\{1, m\} \subseteq M$ by Cases 1–3 and Condition 4. For $i \in M$ such that $i < \max\{M\}$, let $N(i)$ be the the next larger number in M than i. We define $N(\max\{M\}) := \max\{M\} + 1$ for consistency needed later. To estimate the probability for the alternative path A_m, we claim a property for the edges $(v_m, v_{m+1}), \ldots, (v_{N(m)-1}, v_{N(m)})$ in the following lemma.

Lemma 8 *Unless $m = \max\{M\}$, at least one of the edges $(v_m, v_{m+1}), \ldots, (v_{N(m)-1}, v_{N(m)})$ is removed by* `NarrowCut`.

Proof Assume otherwise. If $v_{N(m)} = v_h = p$ and Δ_p is wide, then $v_{N(m)-1} = v_{h-1} \notin U$ and `NarrowCut`$((v_{N(m)}, v_{N(m-1)}) \cup \Delta_{v_{N(m)-1}})$ is performed in `WideEmbedding`(Δ_p). Otherwise, since $\Delta_{v_{N(m)}}$ is narrow and the vertex $v_{N(m)-1}$ is not in the spine of $v_{N(m)}$, `NarrowCut`$((v_{N(m)}, v_{N(m-1)}) \cup \Delta_{v_{N(m)-1}})$ is performed in `NarrowEmbedding`$(\Delta_{v_{N(m)}})$ or `NarrowCut`$((v_{v_{N(m)+1}}, v_{v_{N(m)}}) \cup \Delta_{v_{N(m)}})$. In either case, $(v_{N(m)}, v_{N(m)-1})$ is not removed by assumption. This means that `NarrowCut`$((v_{N(m)-1}, v) \cup \Delta_v)$ is performed for every child v of $v_{N(m)-1}$, but does not remove the edge $(v_{N(m)-1}, v_{N(m)-2})$.

Repeating this argument, `NarrowCut`$((v_m, v) \cup \Delta_v)$ for every child v of v_m is performed. Thus, either the edge (v_m, u_m) is removed, or `NarrowCut`$((v_m, u_m) \cup \Delta_{u_m})$ is performed, contradicting that Condition 4 is satisfied for m. □

For $1 \le i \le \max\{M\}$, let X_i be the probability that the edge (v_i, v_{i-1}) is not removed by `NarrowCut`$((v_i, v_{i-1}) \cup \Delta_{v_{i-1}})$, i.e.,

$$X_i := \left[\frac{w(S_{v_{i-1}})}{w((v_i, v_{i-1})) + w(S_{v_{i-1}})}\right]^2.$$

We define $X_{N(\max\{M\})} = X_{\max\{M\}+1} := 0$.

Lemma 9 *The probability that the vertices v_1 and v_0 are connected not by the edge e but by the alternative path A_m is at most*

$$\frac{2w(e)}{w(D_1)}\left(\prod_{i=2}^{m} X_i\right)\left(1 - \prod_{i=m+1}^{N(m)} X_i\right).$$

Proof We estimate the probabilities for three groups of edges $e = (v_0, v_1)$; (v_1, v_2), …, (v_{m-1}, v_m); and (v_m, v_{m+1}), …, $(v_{N(m)-1}, v_{N(m)})$. First, since e is removed, the probability for this event is

$$\begin{aligned} 1 - X_1 = 1 - \left[\frac{w(S_{v_0})}{w(e) + w(S_{v_0})}\right]^2 &= \frac{w(e)^2 + 2w(e)w(S_{v_0})}{(w(e) + w(S_{v_0}))^2} \\ &\le \frac{2w(e)}{w(e) + w(S_{v_0})} = \frac{2w(e)}{w(D_1)}. \end{aligned}$$

Second, since the edges (v_1, v_2), …, (v_{m-1}, v_m) are not removed (Condition 4), the probability for this event is $\prod_{i=2}^{m} X_i$. Finally, unless $m = \max\{M\}$, at least one of the edges (v_m, v_{m+1}), …, $(v_{N(m)-1}, v_{N(m)})$ is removed by Lemma 8. The probability for this event is $1 - \prod_{i=m+1}^{N(m)} X_i$. If $m = \max\{M\}$, then because we have no edges to be removed, the probability for this event is 1, which is equal to $1 - X_{N(\max\{M\})} = 1 - \prod_{i=m+1}^{N(m)} X_i$.

Putting all this together, the probability for the vertices v_0 and v_1 to be connected not by e but by the alternative path A_m is

$$(1 - X_1)\left(\prod_{i=2}^{m} X_i\right)\left(1 - \prod_{i=m+1}^{N(m)} X_i\right) \le \frac{2w(e)}{w(D_1)}\left(\prod_{i=2}^{m} X_i\right)\left(1 - \prod_{i=m+1}^{N(m)} X_i\right).$$

□

Lemma 10 *The distortion of an edge in E^T is at most $4(4\alpha + 1)\delta + 5$.*

Proof We estimate the distortion of the edge $e = (v_1, v_0)$. Because any edge in E^T is not removed after

`NarrowEmbedding`(Δ_p) or `WideEmbedding`(Δ_p) is finished, by Lemmas 7 and 9, the distortion of e is at most

$$\left[\sum_{m\in M}\frac{2w(e)}{w(D_1)}\left(\prod_{i=2}^{m}X_i\right)\left(1-\prod_{i=m+1}^{N(m)}X_i\right)\mathbb{E}[w(A_m)]+X_1w(e)\right]/w(e)$$
$$\le\frac{2}{w(D_1)}\sum_{m\in M}\left(\prod_{i=2}^{m}X_i-\prod_{i=2}^{N(m)}X_i\right)\{1+(1+4\alpha)\delta\}\,w(D_m)+1 \tag{6}$$

The factor $\sum_{m\in M}\left(\prod_{i=2}^{m}X_i-\prod_{i=2}^{N(m)}X_i\right)w(D_m)$ can be estimated as follows:

$$\sum_{m\in M}\left(\prod_{i=2}^{m}X_i-\prod_{i=2}^{N(m)}X_i\right)w(D_m)$$
$$=\sum_{m\in M}\left(\prod_{i=2}^{m}X_i-\prod_{i=2}^{N(m)}X_i\right)\left[w(D_1)+\sum_{\substack{j\in M\\ j<m}}\left\{w(D_{N(j)})-w(D_j)\right\}\right]$$
$$=\left(\prod_{i=2}^{1}X_i-\prod_{i=2}^{N(\max\{M\})}X_i\right)w(D_1)$$
$$+\sum_{j<\max\{M\}}\sum_{\substack{m\in M\\ m>j}}\left(\prod_{i=2}^{m}X_i-\prod_{i=2}^{N(m)}X_i\right)\left\{w(D_{N(j)})-w(D_j)\right\}$$
$$=w(D_1)+\sum_{\substack{j\in M\\ j<\max\{M\}}}\left(\prod_{i=2}^{N(j)}X_i-\prod_{i=2}^{N(\max\{M\})}X_i\right)\left\{w(D_{N(j)})-w(D_j)\right\}$$
$$=w(D_1)+\sum_{\substack{j\in M\\ j<\max\{M\}}}\left(\prod_{i=2}^{N(j)}X_i\right)\left\{w(D_{N(j)})-w(D_j)\right\}. \tag{7}$$

The factor $\prod_{i=2}^{N(j)}X_i$ for $j<\max\{M\}$ can be estimated as follows:

$$\prod_{i=2}^{N(j)}X_i=\prod_{i=2}^{N(j)}\left[\frac{w(S_{v_{i-1}})}{w((v_i,v_{i-1}))+w(S_{v_{i-1}})}\right]^2$$
$$\le\prod_{i=2}^{N(j)}\left[\frac{w(D_{i-1})}{w((v_i,v_{i-1}))+w(D_{i-1})}\right]^2\quad[\text{by } w(S_{v_{i-1}})\le w(D_{i-1})]$$
$$=\prod_{i=2}^{N(j)}\left[\frac{w(D_{i-1})}{w(D_i)}\right]^2=\frac{w(D_1)^2}{w(D_{N(j)})^2}.$$

Substituting $\prod_{i=2}^{N(j)}X_i$ in (7) with this inequality,

$$\sum_{m\in M}\left(\prod_{i=2}^{m}X_i-\prod_{i=2}^{N(m)}X_i\right)w(D_m)$$
$$\le w(D_1)+\sum_{\substack{j\in M\\ j<\max\{M\}}}\frac{w(D_1)^2}{w(D_{N(j)})^2}\left\{w(D_{N(j)})-w(D_j)\right\}$$
$$\le w(D_1)+\int_{w(D_1)}^{\infty}\frac{w(D_1)^2}{x^2}dx=w(D_1)+\left[-\frac{w(D_1)^2}{x}\right]_{w(D_1)}^{\infty}$$
$$=2w(D_1).$$

Applying this inequality to (6), the distortion of e is at most

$$\frac{2}{w(D_1)}\cdot 2w(D_1)\{1+(1+4\alpha)\delta\}+1=4(4\alpha+1)\delta+5.$$

□

Theorem 1 *Any Halin graph can be embedded into random trees with distortion* $77+32\sqrt{5}\approx 148.6$.

Proof By Lemmas 4, 5, and 10, `EHG` embeds any Halin graph into random trees with distortion $4(4\alpha+1)\delta+5=4(4\alpha+1)\frac{2\alpha}{\alpha-1}+5$. By setting $\alpha:=1+\sqrt{5}/2\approx 2.12$, we obtain the distortion of $77+32\sqrt{5}\approx 148.6$. □

Theorem 2 *Any k-outerplanar graph can be embedded into random trees with distortion* $8\cdot 148.6^{k-1}<149^k$.

Proof Any outerplanar graph can be embedded into random trees with distortion 8 [11]. For $k\ge 2$, by inductively applying Theorem 1 to Lemmas 1 and 2, `Ek0G` embeds any k-outerplanar graph into random trees with distortion $8\cdot 148.6^{k-1}<149^k$. □

Acknowledgments This work was supported by JSPS KAKENHI Grant Number 17K00010.

References

[1] Bartal, Y.: Distributed Paging, *Proc. Dagstul Workshop on On-Line Algorithms* (1996).
[2] Bartal, Y.: Probabilistic Approximation of Metric Spaces and its Algorithmic Applications, *Proc. 37th Annual Symposium on Foundations of Computer Science*, pp. 184–193 (1996).
[3] Bartal, Y.: On Approximating Arbitrary Metrics by Tree Metrics, *Proc. 30th Annual ACM Symposium on Theory of Computing*, pp. 161–168 (1998).
[4] Bartal, Y., Blum, A., Burch, C. and Tomkins, A.: A polylog(n)-Competitive Algorithm for Metrical Task Systems, *Proc. 29th Annual ACM Symposium on Theory of Computing*, pp. 711–719 (1997).
[5] Bartal, Y., Fiat, A. and Rabani, Y.: Competitive Algorithms for Distributed Data Management, *J. Comput. Sys. Sci.*, Vol. 51, No. 3, pp. 341–358 (1995).
[6] Charikar, M., Chekuri, C., Goel, A., GUHA, S. and Plotkin, S.: Approximating a Finite Metric by a Small Number of Tree Metrics, *Proc. 39th Annual Symposium on Foundations of Computer Science*, pp. 379–388 (1998).
[7] Chekuri, C., Gupta, A., Newman, I., Rabinovich, Y. and Sinclair, A.: Embedding k-Outerplanar Graphs into ℓ_1, *SIAM J. Discrete Math.*, Vol. 20, No. 1, pp. 119–136 (2006).
[8] Deza, M. M. and Laurent, M.: *Geometry of Cuts and Metrics*, Springer (1997).
[9] Emek, Y.: k-Outerplanar Graphs, Planar Duality, and Low Stretch Spanning Trees, *Algorithmica*, Vol. 61, No. 1, pp. 141–160 (2011).
[10] Fakcharoenphol, J., Rao, S., and Talwar, K.: A Tight Bound on Approximating Arbitrary Metrics by Tree Metrics, *J. Comput. Syst. Sci.*, Vol. 69, No. 3, pp. 485–497 (2004).
[11] Gupta, A., Newman, I., Rabinovich, Y. and Sinclair, A.: Cuts, Trees, and ℓ_1-Embedding of Graphs, *Combinatorica*, Vol. 24, No. 2, pp. 233–269 (2004).
[12] Halin, R.: Über simpliziale Zerfällungen beliebiger (endlicher oder unendlicher) Graphen, *Math. Ann.*, Vol. 156, No. 3, pp. 216–225 (1964).
[13] Halin, R.: Studies on Minimally n-Connected Graphs, *Combinatorial Mathematics and its Applications (Proc. Conf., Oxford, 1969)*, Academic Press, London, pp. 129–136 (1971).
[14] James R. Lee, A. S.: Pathwidth, Trees, and Random Embeddings, *Combinatorica*, Vol. 33, No. 3, pp. 349–374 (2013).
[15] Linial, N., London, E. and Rabinovich, Y.: The Geometry of Graphs and Some of its Algorithmic Applications, *Combinatorica*, Vol. 15, No. 2, pp. 215–245 (1995).
[16] Lund, C., Reingold, N., Westbrook, J. and Yan, D.: Competitive On-Line Algorithms for Distributed Data Management, *SIAM J. Comput.*, Vol. 28, No. 3, pp. 1086–1111 (1999).
[17] Okamura, H. and Seymour, P. D.: Multicommodity Flows in Planar Graphs, *J. Combin. Theory Ser. B*, Vol. 31, pp. 75–81 (1981).