Upward Point Set Embeddability for Convex Point Sets Is in $P^\star$

Michael Kaufmann[1], Tamara Mchedlidze[2], and Antonios Symvonis[2]

[1] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany
mk@informatik.uni-tuebingen.de
[2] Dept. of Mathematics, National Technical University of Athens, Greece
{mchet,symvonis}@math.ntua.gr

Abstract. In this paper, we present a polynomial dynamic programming algorithm that tests whether a n-vertex directed tree T has an upward planar embedding into a convex point-set S of size n. We also note that our approach can be extended to the class of outerplanar digraphs. This nontrivial and surprising result implies that any given digraph can be efficiently tested for an upward planar embedding into a given convex point set.

1 Introduction

A *planar straight-line embedding* of a graph G into a point set S is a mapping of each vertex of G to a distinct point of S and of each edge of G to the straight-line segment between the corresponding end points so that no two edges cross each other. Planar straight-line embeddings for outerplanar graphs and trees were studied by Gritzmann *et al.* [11], Bose [4] and Bose *et al.* [5]. Cabello [6] proved that the problem to decide whether a given planar graph admits a planar straight-line embedding into a given point set is $\mathcal{NP}$-hard. Planar graph embeddings into point sets, where edges are allowed to bend, have also been studied (see, e.g., [2,7,12,14,17]).

An *upward planar directed graph* is a digraph that admits a planar drawing such that each edge is represented by a curve monotonically increasing in the y-direction. An *upward straight-line embedding* (*UPSE* for short) of an upward planar digraph G into a point set S is a mapping of each vertex of G to a distinct point of S and of each edge to the straight-line segment between its corresponding end points such that no two edges cross and for each edge (u, v) the condition $y(u) < y(v)$ holds, for the y-coordinates $y(u)$ and $y(v)$. *Upward point set embeddability* is the decision problem of whether a given digraph has an UPSE into a given point set.

* This research has been co-financed by EUROGIGA project GraDR 10-EuroGIGA-OP-003 and by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

M. van Kreveld and B. Speckmann (Eds.): GD 2011, LNCS 7034, pp. 403–414, 2012.

Upward point set embeddability was first studied by Giordano et al. [9]. The authors studied the version of the problem where bends on edges are allowed and showed that every planar *st*-digraph admits an upward point set embedding with at most two bends per edge. Upward point set embeddability with a given mapping, i.e., where a correspondence between the nodes and the point set is part of the input, was studied in [10,16]. Recently, straight-line drawings were studied in [1,3,8] and many interesting and partial results were presented. Among them are several results concerning upward point set embeddability of a tree into a convex point set. More specifically, several families of trees were presented, which have an UPSE into every convex point set, i.e., caterpillars, switch-trees, hourglass trees. On the other hand, it was demonstrated that the family of k-switch trees (generalization of switch-trees) does not have an UPSE into all convex point sets. An immediate question that arises from these facts is whether the existence of an UPSE of a tree into a convex point set can be efficiently tested. The contribution of this paper is an affirmative answer to this question. More specifically, we show that, given a directed tree T and a convex point set S, it can be tested in polynomial time whether T has an UPSE into S.

Recently, Geyer *et al.* [8] proved that the general *upward point-set embeddability* problem is $\mathcal{NP}$-complete even for m-convex point sets[1]. Thus one interesting open problem regarding UPSE was whether there exists a class of upward planar digraphs $\mathcal{D}$ for which the upward point set embeddability problem remains $\mathcal{NP}$-complete even for convex point sets. We answer this question in the negative by extending our UPSE algorithm for trees to the class of outerplanar graphs. Since any graph admitting a planar embedding into a convex point set is an outerplanar digraph, our result implies that the upward point-set embeddability can be efficiently solved for convex point sets and general digraphs.

For simplicity of presentation, we concentrate on the case of directed trees. In Section 2, we present the necessary notation and some basic results on UPSE, which are utilized by our tree algorithm. In Section 3, we study a restricted version of the UPSE problem which fixes the point in which the root of the tree is embedded and places restrictions on the drawing of subtrees. In Section 4, we explore the result of Section 3 and present a dynamic programming algorithm for deciding whether a directed tree has an UPSE into a convex point set. In Section 5 we state the extended result for outerplanar digraphs. Due to space constraints the proof of this result as well as some other proofs are presented in the extended version of this paper [15].

2 Notation - Preliminaries

Point Sets. Let S be a set of points on the plane. We assume that the points of S are in general position, i.e., no three of them lie on the same line. Moreover, we also assume that no two points of S share the same y-coordinate; if they do, a slight rotation of the coordinate axes can ensure that all points have distinct

[1] An m-convex point set can be intuitively defined as a set of m shelled, one into another, distinct convex point sets.

y-coordinates. The *convex hull* $CH(S)$ of S is the point set that is obtained as a convex combination of the points of S. A point set such that no point is in the convex hull of the others is called a *point set in convex position*, or a *convex point set*. Given a point set S, by $t(S)$ (resp., $b(S)$) we denote the top (bottom) point of S i.e., the point with the largest (resp., smallest) y-coordinate.

A *one-sided convex point set* S is a convex point set in which $b(S)$ and $t(S)$ are adjacent on the border of $CH(S)$. If $t(S)$ and $b(S)$ appear adjacent and in this order on the border of $CH(S)$ as we traverse it in the clockwise (resp., counterclockwise) direction, then the one-sided convex point set is called a *left-sided convex point set* (resp., *right-sided convex point set*). A point set consisting of at most two points is considered to be either a left-sided or a right-sided convex point set. A convex point set which is not one-sided, is called a *two-sided convex point set*.

Each given convex point set S may be considered to be the union of two specified (at the time S is given) one-sided convex point sets, one left-sided which is denoted by $L(S)$ and is referred to as the *left-side* of S, and one right-sided which is denoted by $R(S)$ and is referred to as the *right-side* of S. When there is no confusion regarding the point set S we refer to, for simplicity, we use the terms L and R instead of $L(S)$ and $R(S)$, respectively. Each of the points $b(S)$ and $t(S)$ belongs to either $L(S)$ or $R(S)$ but not both.

A subset of points of a convex point set S is called *consecutive* if its points appear consecutively as we traverse the convex hull of S in clockwise direction. Given that all points of S have distinct y-coordinates, we can refer to the first, the second, the third, etc., lowest point on the left (right) side of S. By $p_i^L,\ 1 \leq i \leq |L(S)|$, we denote the i-th lowest point on the left side of S. Similarly, by $p_i^R,\ 1 \leq i \leq |R(S)|$, we denote the i-th lowest point on the right side of S.

Let $S_{a..b,c..d} = \{p_i^L \mid a \leq i \leq b\} \cup \{p_i^R \mid c \leq i \leq d\}$ denote the subset of S consisting of $b-a+1$ consecutive points on the left side of S, starting from point p_a^L in the clockwise direction, and of $d-c+1$ consecutive points on the right side, starting from point p_c^R in the counterclockwise direction. For simplicity, for a one-sided point set S we use the notation $S_{a..b}$.

In this paper, we assume that queries of the form *"Find the i-th point on the left/right side of the convex point set S"* can be answered in $O(1)$ time, e.g., the points on each side of S are stored in an array in ascending order of their y-coordinates.

Trees. Consider a *directed tree* T, i.e., a directed acyclic graph whose underlying undirected structure is that of a tree. Tree T is *rooted* if one of its vertices, denoted by $r(T)$, is designated as its *root*. We then say that T *is rooted at* vertex $r(T)$. By $d^-(v)$ (resp., $d^+(v)$) we denote the in-degree (resp., the out-degree) of vertex v of T. By $d(v)$ we denote the total degree of vertex v, i.e., $d(v) = d^-(v) + d^+(v)$.

Let T be a rooted tree and let $r = r(T)$ be its root. Let $T_1^l, \ldots, T_{d^-(r)}^l, T_1^h, \ldots, T_{d^+(r)}^h$ be the rooted subtrees of T obtained by removing from T its root r and r's incident arcs and having as their roots the vertices that are incident to r by either an incoming or an outgoing arc (see Figure 1.a). Trees $T_1^l, \ldots, T_{d^-(r)}^l$,

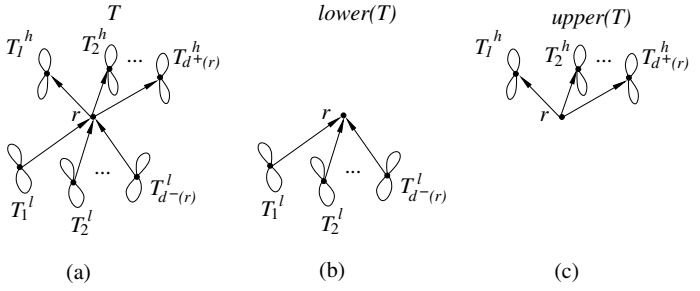

Fig. 1. (a) A rooted at vertex r tree T and its subtrees $T_1^l, \ldots, T_{d^-(r)}^l$, $T_1^h, \ldots, T_{d^+(r)}^h$. (b) The subtree $lower(T)$ of T. (c) The subtree $upper(T)$ of T.

$T_1^h, \ldots, T_{d^+(r)}^h$ are called the *subtrees of* T. Note that the superscripts "l" and "h" indicate whether a particular subtree of T is connected to r by an incoming to r or by an outgoing from r arc, respectively.

The rooted subtree of T consisting of T's root, r, together with $T_1^l, \ldots, T_{d^-(r)}^l$ is called the *lower subtree of* T and is also rooted at r. The lower subtree of T is denoted by $lower(T)$ (Figure 1.b). Similarly, the rooted subtree of T consisting of T's root, r, together with $T_1^h, \ldots, T_{d^+(r)}^h$ is called the *upper subtree of* T and is also rooted at r. The upper subtree of T is denoted by $upper(T)$ (Figure 1.c).

In this paper, we use the notation $\{u, v\}$ to denote arc (u, v) if $(u, v) \in T$ or arc (v, u) if $(v, u) \in T$. If u is mapped to point p and v is mapped to point q that is located below p, then we say that $\{u, v\}$ is drawn upward (downward) if $(v, u) \in T$ ($(u, v) \in T$).

2.1 Some known Results on UPSE of Rooted Directed Trees

We present some known results on UPSE of rooted directed trees that will be utilized by our algorithms. Binucci *et al.*[3] proved the following lemma concerning the placement of the subtrees of T in an UPSE of T on a convex point set.

Lemma 1 (Binucci *et al.* [3]). *Let T be a n-vertex directed tree rooted at r and let S be any convex point set of size n. Let $T_1, T_2, \ldots, T_{d(r)}$ be the subtrees of T. Then, in any UPSE of T into S, the vertices of subtree T_i are mapped to a set of consecutive points of S, $1 \leq i \leq d(r)$.* □

The following lemma concerns the UPSE of a rooted tree into a *one-sided* convex point set. It can be considered to be a simple restatement of a result by Heath *et al.* [13] (Theorem 2.1).

Lemma 2. *Let T be a n-vertex directed tree rooted at r and S be a one-sided convex point set of size n. Let $T_1, T_2, \ldots, T_{d(r)}$ be the subtrees of T. Then, T admits an UPSE into S so that the following are true:*

i) Each T_i, $1 \leq i \leq d(r)$, is drawn on consecutive points of S.

ii) If the root r of T is mapped to point p_r then there is no arc connecting a point of S below p_r to a point of S above p_r.

By utilizing Lemma 2, we prove the following.

Lemma 3. *Let T be a n-vertex directed tree rooted at r and S be a one-sided convex point set of size n. Then, an UPSE of T into S satisfying the properties of Lemma 2 can be obtained in $O(n)$ time. Moreover, after $O(n)$ time preprocessing, the point p_r that hosts the root r of T can be determined in $O(1)$ time (i.e., without determining the complete UPSE of T into S).*

Proof. Let $k = |lower(T)|$ be the size of subtree $lower(T)$ (rooted at r). Assuming that T was preprocessed in $O(n)$ time, k can be retrieved in constant time. It immediately follows that in an UPSE of T into S satisfying the properties of Lemma 2 there are $k-1$ vertices of T (all belonging to $lower(T)$) that are placed below r. Thus, r is mapped to the k-th lowest point of S. This point, say p_r, can be computed in $O(1)$ time. Having decided where to place the root r, the UPSE of T can be completed in $O(n)$ time by recursively embedding the vertices of $lower(T)$ ($upper(T)$) to the points of S below (above) p_r. □

3 A Restricted UPSE Problem for Rooted Directed Trees

In this section, we study a restricted UPSE problem that will be later on used by our main algorithm which decides whether there exists an UPSE of a given directed tree into a given convex point set.

Definition 1. *In a* restricted UPSE *problem for trees we are given a directed tree T rooted at r, a convex point set S, and a point $p_r \in S$. We are asked to decide whether there exists an UPSE of T into S such that (i) the root r of T is mapped to point p_r and, (ii) each subtree of T (rooted at r) is mapped to consecutive points on the same side (either L or R) of S.*

The following observation follows directly from the above definition.

Observation 1. *In a restricted UPSE of a directed tree T rooted at r into a convex point set $S = L \cup R$, where the root r of T is mapped to point $p_r \in S$, no edge enters the triangles $\triangle(t(L), t(R), p_r)$ and $\triangle(b(L), b(R), p_r)$.*

Figure 2.a shows a tree T rooted at vertex r, a convex point set S consisting of a left-sided convex point set L and a right-sided convex point set R. Tree T has a restricted UPSE only if its root r is mapped to point $p_r \in L$ (Figure 2.b). Mapping r to any other point $p \in S$ makes it impossible to map each subtree of T to consecutive points on the same side of S.

Before we proceed to describe a decision algorithm for the restricted UPSE problem, we need some more notation. Let T be a directed tree rooted at vertex r and let $\lambda = (T_1, \ldots, T_{d(r)})$ be an ordering of the subtrees of T. Let S be a convex point set and let Γ be an UPSE of T into S. We say that UPSE Γ *respects ordering* λ if for any two subtrees T_i and T_j, $1 \leq i \leq j \leq d(r)$, that are

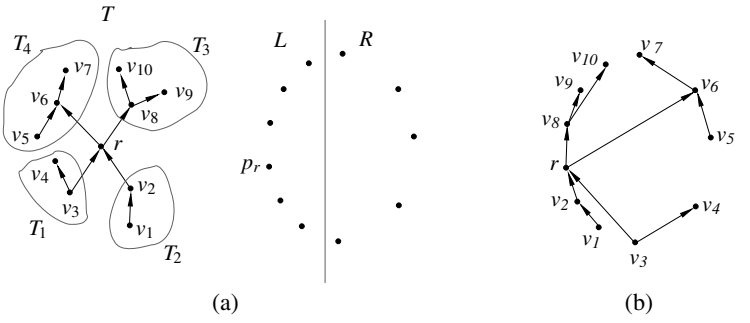

Fig. 2. (a) A tree T rooted at vertex r and a convex point set $S = L \cup R$. (b) A restricted UPSE of T into S so that r is mapped to point p_r. No restricted UPSE of T exists when r is mapped to any point other than p_r.

both mapped on the same side of S, T_i is mapped to a point set that is entirely below the point set T_j is mapped to.

Consider a tree T rooted at vertex r and let $\lambda = (T^l_1, \ldots, T^l_{d^-(r)}, T^h_1, \ldots, T^h_{d^+(r)})$ be an ordering of the subtrees of T. Ordering λ is called a *proper ordering* of the subtrees of T if it satisfies the following properties:

(i) $|upper(T^l_i)| \leq |upper(T^l_j)|,\ 1 \leq i \leq j \leq d^-(r)$, and
(ii) $|lower(T^h_i)| \geq |lower(T^h_j)|,\ 1 \leq i \leq j \leq d^+(r)$.

In Figure 2.a, ordering $\lambda_1 = (T_2, T_1, T_4, T_3)$ is a proper ordering of the subtrees of T, while ordering $\lambda_2 = (T_1, T_2, T_3, T_4)$ is not. Observe that in a proper ordering λ of T, the subtrees in the lower subtree of T appear before the subtrees in the upper subtree of T. The following lemma can be proved by reconstruction.

Lemma 4. *Let T be a n-vertex directed tree rooted at vertex r, λ be a proper ordering of the subtrees of T, and S be a convex point set of size n. Then, if there exists a restricted UPSE of T into S, there also exists a restricted UPSE of T into S that respects λ.* □

Theorem 1. *Let T be a n-vertex directed tree rooted at vertex r, L and R be left-sided and right-sided convex point sets, resp., such that $S = L \cup R$ is a convex point set of size n, and p_r a point of S. The restricted UPSE problem with input T, S and p_r can be decided in $O(d(r)n)$ time. Moreover, if a restricted UPSE for T, S and p_r exists, it can also be constructed in $O(d(r)n)$ time.*

Proof. Let $\lambda = (T_1, T_2, \ldots, T_{d(r)})$ be a proper ordering of the subtrees of T. Proper ordering λ can be computed in $O(n)$ time by a simple tree traversal that computes at the root of T the number of vertices in each subtree of $T \setminus \{v\}$ followed by a bucket sort of the sizes of the subtrees rooted at r. Since the restricted UPSE problem will be repeatedly solved on subtrees of T, we assume that T has been appropriately preprocessed in $O(n)$ time and, thus, a proper ordering of these subtrees can be then computed in $O(d(r))$ time. By Lemma 4,

it is enough to test whether there exists a restricted UPSE that respects λ. Thus, we will describe a dynamic programming algorithm that tests whether there exists a restricted UPSE on input T, L, R and p_r.

Our dynamic programming algorithm uses a two-dimensional $d(r) \times |L|$ matrix M. Value $M[i, j]$ is $TRUE$ if and only if there exists a restricted UPSE of the subtree of T induced by r and $T_1, \ldots, T_i$ that uses all the j lowest points of the left-sided point set L and as many consecutive points as required in the lowest part of the right-sided convex point set R. Recall that $\{u, v\}$ denotes arc (u, v) if $(u, v) \in T$; arc (v, u) if $(v, u) \in T$; otherwise it is undefined.

For the boundary conditions of our dynamic programming we have that:

$$M[0,0] = TRUE$$

$$M[1,j] = \begin{cases} TRUE, & \text{if } j = 0 \text{ and } p_r \notin R_{1..|T_1|} \text{ and } \{r(T_1), p_r\} \text{ is upward} \\ TRUE, & \text{if } j = |T_1| \text{ and } p_r \notin L_{1..|T_1|} \text{ and } \{r(T_1), p_r\} \text{ is upward} \\ FALSE, & \text{otherwise} \end{cases}$$

Let $\sigma = |T_1| + \ldots + |T_i|$. Value $M[i, j]$, $1 < i \leq d(r)$ and $0 \leq j \leq |L|$, is set to $TRUE$ if any of the following conditions is true; otherwise it is set to $FALSE$.

c-1: $M[i, j-1] = TRUE$ ***and*** $p_r = L_{j..j}$.
This is the case where point p_r happens to be the j-th point of L. There is no need to test for upwardness of $\{r(T_i), p_r\}$ since it has been already tested when entry $M[i, j-1]$ was filled in.

c-2: $M[i-1, j-|T_i|] = TRUE$ ***and*** $p_r \notin L_{j-|T_i|+1..j}$ ***and*** $\{r(T_i), p_r\}$ ***is upward***.
In this case, T_i is placed on L. We know that T_i fits on L since $j < |L|$, however, we must make sure that it also holds that p_r is not one of the $|T_i|$ topmost points of $L_{1..j}$.

c-3: $M[i-1, j] = TRUE$ ***and*** $p_r \in R_{1..\sigma-j-|T_i|+1}$ ***and*** $\sigma - j + 1 \leq |R|$ ***and*** $\{r(T_i), p_r\}$ ***is upward***.
In this case, T_i is placed to R. If p_r is one of the points in $R_{1..\sigma-j-|T_i|+1}$ then we have to make sure that at least $\sigma - j + 1$ points exist in $|R|$.

c-4: $M[i-1, j] = TRUE$ ***and*** $p_r \notin R_{1..\sigma-j}$ ***and*** $\sigma - j \leq |R|$ ***and*** $\{r(T_i), p_r\}$ ***is upward***.
In this case, T_i is also placed to R. However, in contrast to case **c-3**, p_r is not one of the points in $R_{1..\sigma-j}$. Thus, we only need to make sure that at least $\sigma - j$ points exist in $|R|$.

When determining the value of an entry $M[i, j]$ we need to decide whether arc $\{r(T_i), p_r\}$ is upward. In order to do that, we need to know the point to which $r(T_i)$ is mapped. By Lemma 3, this point can be computed in $O(1)$ time since T_i is mapped to $|T_i|$ consecutive points forming a one-sided convex point set.

It can be easily verified that entry $M[d(r), |L|] = TRUE$ if and only if there is a restricted UPSE of T into $L \cup R$ such that $r(T)$ is mapped to p_r.

Each entry of matrix M can be filled in $O(1)$ time. Thus, all entries of matrix M are filled in $O(d(r)|L|)$ time. The embedding, if exists, can be constructed by storing in each entry $M[i, j]$ (that was set to $TRUE$) the side ("L" or "R") in

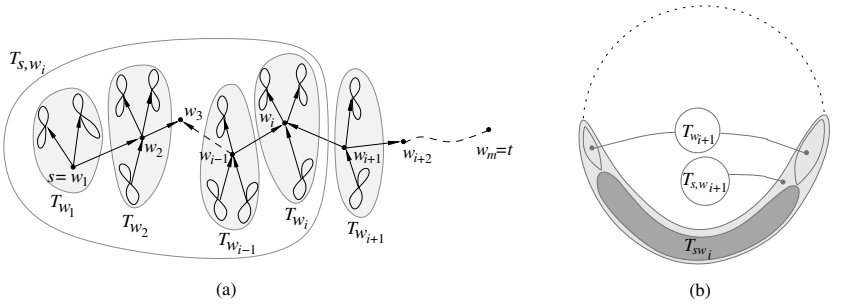

Fig. 3. (a) The decomposition of tree T based on a path between a source s and a sink t of T. (b) The structure of an UPSE of the tree T into point set S.

which T_i was placed. This information, together with the fact that the restricted UPSE respects ordering λ is sufficient to construct the embedding. □

Denote by $\mathcal{L}(T, L, R)$ the set of points $p \in L \cup R$ such that there exists a restricted UPSE of T into $L \cup R$ where the root of T is mapped to p. The next theorem follows from Theorem 1, testing each point of $L \cup R$ as a candidate host for $r(T)$.

Theorem 2. *Let T be an n-vertex directed tree rooted at vertex r and L and R be left-sided and right-sided convex point sets, resp., such that $S = L \cup R$ is a convex point set of size n. Set $\mathcal{L}(T, L, R)$ can be computed in $O(d(r)n^2)$ time.* □

4 The Testing Algorithm for Directed Trees

Let T be a directed tree and let S be a convex point set. In any UPSE of T into S, a source node s and a sink node t of T will be mapped to points $b(S)$ and $t(S)$, respectively. In this section, we present a dynamic programming algorithm that decides in polynomial time whether, given a n-vertex directed tree T, a source s and a sink t of T, and a convex point set S of size n, T has an UPSE into S so that s and t are mapped to $b(S)$ and $t(S)$, respectively. Applying this algorithm on all $\langle source, sink \rangle$ pairs of T, yields a polynomial time algorithm for deciding whether T has an UPSE into S.

Let s and t be a source and a sink vertex of T, respectively. Denote by $P_{s,t} = \{s = w_1, w_2, \ldots, w_m = t\}$ the (undirected) path connecting s and t in T, see Figure 3.a. By T_{s,w_i}, $1 \leq i < m$, we denote the subtree of T that contains source s and is formed by the removal of edge $\{w_i, w_{i+1}\}$. By definition, $T_{s,w_m} = T$. Let $T_{w_i} = T_{s,w_i} \setminus T_{s,w_{i-1}}$, $1 < i \leq m$. By definition, $T_{w_1} = T_{s,w_1}$. By Lemma 1, we know that T_{s,w_i} is drawn on consecutive points of S, call this point set S_i (see also Figure 3.b). Since s is mapped to $b(S)$, we infer that $b(S) \in S_i$. Similarly, in any UPSE of T into S, $T_{s,w_{i+1}}$ is also drawn on consecutive points of S that contain $b(S)$, call this point set S_{i+1}. Hence, $T_{w_{i+1}}$ is drawn on a set $S_{w_{i+1}} = S_{i+1} \setminus S_i$, that is, a subset of S comprised by two consecutive point sets of S, one on its left and one on its right side.

Our dynamic programming algorithm maintains a list of points $\mathcal{P}(a,b,k),\ 0 \leq a \leq |L|,\ 0 \leq b \leq |R|,\ 1 \leq k \leq m$, such that:

$$p \in \mathcal{P}(a,b,k) \Longleftrightarrow \begin{cases} T_{s,w_k} \text{ has an UPSE into point set } S_{1..a,1..b} \text{ with} \\ \text{vertex } w_k \text{ mapped to point } p. \end{cases} \quad (1)$$

For the boundary conditions of our dynamic programming we have that $\mathcal{P}(a,b,1) = \mathcal{L}(T_{w_1}, L_{1..a}, R_{1..b})$ where $a + b = |T_{w_1}|$. Note that since w_1 is a source, $\mathcal{P}(a,b,1)$ is either $\{b(s)\}$ or $\emptyset$.

Our dynamic programming is based on the following recurrence relation, which allows us to add points in $\mathcal{P}(a,b,i)$. For any $1 < i \leq m$ we set:

$$\begin{aligned} \mathcal{P}(a,b,i) = \{p \ \mid\ & \exists a_1, b_1 \in Z : a_1 + b_1 = |T_{w_i}| \\ & \textbf{and } p \in \mathcal{L}(T_{w_i}, L_{a-a_1+1..a}, R_{b-b_1+1..b}) \\ & \textbf{and } \exists q \in \mathcal{P}(a-a_1, b-b_1, i-1) \\ & \textbf{and } \{p,q\} \text{ is upward} \quad \} \end{aligned} \quad (2)$$

Next we prove that the recurrence relation (2) satisfies the property described by equivalence (1). We start with the forward direction. From the boundary conditions it is true for $i = 1$. Assume that if $q \in \mathcal{P}(a-a_1, b-b_1, i-1)$ then $T_{s,w_{i-1}}$ has an UPSE into $S_{1..a-a_1,1..b-b_1}$ with vertex w_{i-1} mapped to point q. Let now $p \in \mathcal{P}(a,b,i)$. By the definition of the recurrence relation we infer that: (1) there exist $a_1, b_1 \in Z$ so that $a_1 + b_1 = |T_{w_i}|$, (2) $p \in \mathcal{L}(T_{w_i}, L_{a-a_1+1..a}, R_{b-b_1+1..b})$, which by definition of $\mathcal{L}$, means that there is a restricted UPSE of T_{w_i} into $L_{a-a_1+1..a}, R_{b-b_1+1..b}$ with w_i mapped to p, (3) $\exists q \in \mathcal{P}(a-a_1, b-b_1, i-1)$, thus, by induction hypothesis, $T_{s,w_{i-1}}$ has an UPSE into $S_{1..a-a_1,1..b-b_1}$, and, finally, (4) edge $\{p,q\}$ is upward. Then we combine the UPSE for $T_{s,w_{i-1}}$ with the restricted UPSE for T_{w_i} in order to get an UPSE of T_{s,w_i} on point set $S_{1..a,1..b}$. By Observation 1, we have that the combined drawing is planar.

For the reversed statement we also work by induction. From the boundary conditions we know that if $T_{s,w_1} = T_{w_1}$ has an UPSE into a point set $S_{1..a,1..b}$ then $b(S) \in \mathcal{P}(a,b,1)$, where $a + b = |T_{w_1}|$. Assume that the statement is true for $T_{s,w_{i-1}}$, i.e., if $T_{s,w_{i-1}}$ has an UPSE into a point set $S_{1..a,1..b}$ with vertex w_{i-1} mapped to q then $q \in \mathcal{P}(a,b,i-1)$. Assume also that T_{s,w_i} has an UPSE into a point set $S_{1..a,1..b}$ with vertices s and w_i mapped to points $b(S)$ and p, respectively. By the discussion above we know that in every such embedding $T_{s,w_{i-1}}$ is mapped to consecutive points of $S_{1..a,1..b}$ that contains $b(S)$. Therefore there exist two numbers a_1 and b_1, so that $a_1 + b_1 = |T_{w_i}|$ and subtree T_{w_i} is mapped to the point set $S_{a-a_1+1..a,b-b_1+1..b}$, with vertex w_i mapped to some point p, $p \in S_{a-a_1+1..a,b-b_1+1..b}$. Moreover, by induction hypothesis, there exists $q \in \mathcal{P}(a-a_1, b-b_1, i-1)$. So, since the edge connecting p and q is upward, by the definition of recurrence relation we infer that $p \in \mathcal{P}(a,b,i)$.

Finally we note that, an UPSE of T into S such that source s and sink t are mapped to $b(S)$ and $t(S)$, respectively, exists if and only if $\mathcal{P}(|L|,|R|,m)$ is non-empty. Note that if $\mathcal{P}(|L|,|R|,m) \neq \emptyset$, then it must hold that $\mathcal{P}(|L|,|R|,m) = \{t(S)\}$. The values $\mathcal{P}(a,b,k)$, when $0 \leq a \leq |L|$, $0 \leq b \leq |R|$, $1 \leq k \leq m$ are calculated by Algorithm 1.

Algorithm 1. TREE-UPSE(T, S, s, t)

input : A directed tree T, a point set S, a source s and a sink t of T. Path $(s = w_1, \ldots, w_m = t)$ is used to progressively build tree T from subtrees $T_{w_i},\ 1 \leq i \leq m$.

output : "YES" if T has an UPSE into S with s mapped to $b(S)$ and t mapped to $t(S)$, "NO" otherwise.

1. **For** $a = 0 \ldots |L|$
2. **For** $b = 0 \ldots |R|$
3. $\mathcal{P}(a, b, 1) = \mathcal{L}(T_{w_1}, L_{1..a}, R_{1..b})$
4. **For** $k = 2 \ldots m$ //Consider tree T_{w_k}
5. $\mathcal{P}(a, b, k) = \emptyset$
6. **For** $i = 0 \ldots |T_{w_k}|$ //We consider the case where i vertices of T_{w_k} are placed to the left side of S
7. **if** $(a - i \geq 0)$ **and** $(b - (|T_{w_k}| - i) \geq 0)$
8. Let $\mathcal{L} = \mathcal{L}(T_{w_k}, L_{a-i+1..a}, R_{b-(|T_{w_k}|-i)+1..b})$
9. //We consider all possible placements of w_{k-1}
10. **For** each q in $\mathcal{P}(a - i, b - (|T_{w_k}| - i), k - 1)$
11. //We consider all the possible placements of vertex w_k
12. **For** each p in $\mathcal{L}$
13. **if** ($\{w_{i-1}, w_i\}$ drawn on line-segment (q, p) is upward)
14. **then** add p to $\mathcal{P}(a, b, k)$.
15. **if** $\mathcal{P}(|L|, |R|, m)$ is empty **then return**("NO");
16. **return**("YES");

Theorem 3. *Let T be a n-vertex rooted directed tree, S be a convex point set of size n, s be a source of T and t be a sink of T. It can be decided in time $O(n^5)$ whether T has an UPSE on S such that s is mapped to $b(S)$ and t is mapped to $t(S)$. If such an UPSE exists, it can be constructed within the same time bound.*

Proof. A naive analysis of Algorithm 1 yields an $O(n^7)$ time complexity. The analysis assumes that (i) the left and the right side of S have both size $O(n)$, (ii) the path from s to t has length $O(n)$, (iii) each tree T_{w_i} has size $O(n)$ and (iv) each $\mathcal{L}$-list containing the solution of a restricted UPSE problem is computed in $O(n^3)$ time. However, based on the following two observations, the total time complexity can be reduced to $O(n^5)$.

A factor of n can be saved by realizing that in our dynamic programming we can maintain a list $\mathcal{P}'(a, i)$ which uses only one parameter for the left side of the convex set (in contrast with $\mathcal{P}(a, b, i)$ which uses a parameter for each side of S). The number of points on the right side of S is implied since the size of each tree T_{s,w_i} is fixed. For simplicity, we have decided to use notation $\mathcal{P}(a, b, i)$. Another factor of n can be saved by observing that the solution of a restricted UPSE is actually $O(deg(w_i)n^2)$. Thus, summing over all i gives $O(n^3)$ in total, and not $O(n^4)$.

The UPSE of T into S can be recovered easily by modifying Algorithm 1 so that it stores for each point $p \in \mathcal{P}(a, b, k)$ the point q where vertex w_{i-1} is mapped to as well as the point set that hosts tree $T_{s,w_{i-1}}$ (i.e., its top point on the left and the right side of S). □

By applying Algorithm 1 on all $\langle source, sink \rangle$ pairs of T we can decide whether tree T has an UPSE on a convex point set S, as the main next theorem indicates.

Theorem 4. *Let T be a n-vertex rooted directed tree and S be a convex point set of size n. It can be decided in time $O(n^6)$ whether T has an UPSE into S. If such an UPSE exists, it can also be constructed within the same time bound.*

Proof. Note that a naive application of the idea leads to the algorithm with time complexity $O(n^7)$, since there are $O(n^2)$ distinct pairs of sources and sinks. Next we explain how the overall time complexity can be reduced to $O(n^6)$. Let $P_{s,t}$ be a path from s to t, passing through m vertices, and let t' be the j-th vertex of $P_{s,t}$ that is also a sink of G. During the computation of $\mathcal{P}(a, b, m)$ corresponding to path $P_{s,t}$ we also compute $\mathcal{P}(a, b, j)$ and thus we can immediately answer whether there exists an UPSE of G into S so that s and t' is mapped to $b(S)$ and $t(S)$, respectively. Next consider a sink $\tilde{t}$ that does not belong to path $P_{s,t}$. Consider the path $P_{s,\tilde{t}}$. Assume that the last common vertex of $P_{s,t}$ and $P_{s,\tilde{t}}$ is the j-th vertex of $P_{s,t}$. In order to compute whether there is an UPSE of G into S so that s and $\tilde{t}$ are mapped to $b(S)$ and $t(S)$, respectively, we can start the computations of Algorithm 1 determined by variable k from the $j + 1$-th step (see line 4 of the algorithm). Thus, for a single source s and all possible sinks variable k changes at most n times. Since the number of different sources is $O(n)$ we conclude that the whole algorithm runs in time $O(n^6)$. □

5 Conclusions

In this paper we presented a polynomial dynamic programming algorithm that tests whether a n-vertex directed tree T has an upward planar embedding into a convex point-set S of size n. In the long version of this paper [15] we explain how our approach can be extended to the class of outerplanar digraphs, obtaining the following theorem.

Theorem 5. *Let G be a n-vertex digraph and S be a convex point set of size n. It can be decided in polynomial time whether G has an UPSE into S. Moreover, if such an UPSE exists, it can also be constructed in polynomial time.* □

Acknowledgments. We thank Markus Geyer for the useful discussions during the work on this paper.

References

1. Angelini, P., Frati, F., Geyer, M., Kaufmann, M., Mchedlidze, T., Symvonis, A.: Upward Geometric Graph Embeddings into Point Sets. In: Brandes, U., Cornelsen, S. (eds.) GD 2010. LNCS, vol. 6502, pp. 25–37. Springer, Heidelberg (2011)
2. Badent, M., Di Giacomo, E., Liotta, G.: Drawing colored graphs on colored points. Theor. Comput. Sci. 408(2-3), 129–142 (2008)

3. Binucci, C., Di Giacomo, E., Didimo, W., Estrella-Balderrama, A., Frati, F., Kobourov, S., Liotta, G.: Upward straight-line embeddings of directed graphs into point sets. Computat. Geom. Th. Appl. 43, 219–232 (2010)
4. Bose, P.: On embedding an outer-planar graph in a point set. Computat. Geom. Th. Appl. 23(3), 303–312 (2002)
5. Bose, P., McAllister, M., Snoeyink, J.: Optimal algorithms to embed trees in a point set. J. Graph Alg. Appl. 1(2), 1–15 (1997)
6. Cabello, S.: Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. J. Graph Alg. Appl. 10(2), 353–366 (2006)
7. Di Giacomo, E., Didimo, W., Liotta, G., Meijer, H., Trotta, F., Wismath, S.K.: k-colored point-set embeddability of outerplanar graphs. J. Graph Alg. Appl. 12(1), 29–49 (2008)
8. Geyer, M., Kaufmann, M., Mchedlidze, T., Symvonis, A.: Upward Point-Set Embeddability. In: Černá, I., Gyimóthy, T., Hromkovič, J., Jefferey, K., Králović, R., Vukolić, M., Wolf, S. (eds.) SOFSEM 2011. LNCS, vol. 6543, pp. 272–283. Springer, Heidelberg (2011)
9. Giordano, F., Liotta, G., Mchedlidze, T., Symvonis, A.: Computing Upward Topological Book Embeddings of Upward Planar Digraphs. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 172–183. Springer, Heidelberg (2007)
10. Giordano, F., Liotta, G., Whitesides, S.: Embeddability Problems for Upward Planar Digraphs. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 242–253. Springer, Heidelberg (2009)
11. Gritzmann, P., Mohar, B., Pach, J., Pollack, R.: Embedding a planar triangulation with vertices at specified positions. Amer. Math. Mont. 98, 165–166 (1991)
12. Halton, J.: On the thickness of graphs of given degree. Inf. Sci. 54, 219–238 (1991)
13. Heath, L.S., Pemmaraju, S.V., Trenk, A.N.: Stack and queue layouts of directed acyclic graphs: Part I. SIAM J. Comput. 28(4), 1510–1539 (1999)
14. Kaufmann, M., Wiese, R.: Embedding vertices at points: Few bends suffice for planar graphs. J. Graph Alg. Appl. 6(1), 115–129 (2002)
15. Kaufmann, M., Mchedlidze, T., Symvonis, A.: Upward point set embeddability for convex point sets is in P. Technical report. arXiv:1108.3092, `http://arxiv.org/abs/1108.3092`
16. Mchedlidze, T., Symvonis, A.: On ρ-Constrained Upward Topological Book Embeddings. In: Eppstein, D., Gansner, E.R. (eds.) GD 2009. LNCS, vol. 5849, pp. 411–412. Springer, Heidelberg (2010)
17. Pach, J., Wenger, R.: Embedding planar graphs at fixed vertex locations. Graphs and Combinatorics 17(4), 717–728 (2001)