A Simple Linear Time Algorithm for the Locally Connected Spanning Tree Problem on Maximal Planar Chordal Graphs ☆

Tiziana Calamoneri[a], Matteo Dell'Orefice[a], Angelo Monti[a]

[a] *Computer Science Department, "Sapienza" University of Rome, Italy*

Abstract

A locally connected spanning tree (LCST) T of a graph G is a spanning tree of G such that, for each node, its neighborhood in T induces a connected subgraph in G. The problem of determining whether a graph contains an LCST or not has been proved to be NP-complete, even if the graph is planar or chordal. The main result of this paper is a simple linear time algorithm that, given a maximal planar chordal graph, determines in linear time whether it contains an LCST or not, and produces one if it exists. We give an analogous result for the case when the input graph is a maximal outerplanar graph.

Keywords: locally connected spanning tree, partial k trees, SC k-trees, 2-trees, chordal graphs, planar graphs.

1. Introduction

A *locally connected spanning tree (LCST)* T of a graph G is a spanning tree of G such that for each node its neighborhood in T induces a connected subgraph in G [3]. It is well known that an interconnection network can be modeled as a graph and, in this context, the existence of such a spanning

☆ An extended abstract of this paper has been presented at ICTCS 2016. The work has been partially supported by the Italian Ministry of Education and University, PRIN project "AMANDA: Algorithmics for MAssive and Networked DAta" and by Sapienza University of Rome.

Email addresses: calamo@di.uiroma1.it (Tiziana Calamoneri), matteodellorefice@gmail.com (Matteo Dell'Orefice), monti@di.uiroma1.it (Angelo Monti)

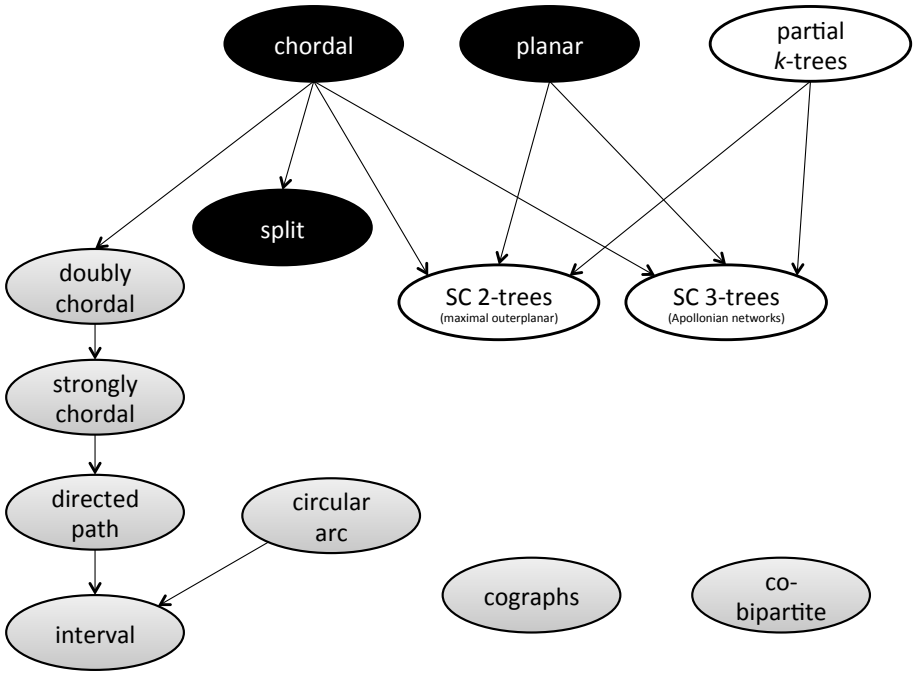

Figure 1: The state of the art concerning the complexity of the LCST problem. The problem is NP-complete on black classes and linearly solvable on grey classes. White classes are studied in this paper, and for them a linear time algorithm is designed.

tree ensures, in case of site failures, effective communication among operative sites as long as these failures are isolated.

Cai proved in [4] that the problem of determining whether a graph contains an LCST (*LCST problem*) is NP-complete even when the input graph is restricted to be planar or split (and, *a fortiori*, chordal). So, researchers have looked for special classes of graphs for which the problem is polynomially solvable.

In particular, in [4] the problem has been proven to admit a linear solution on directed path graphs, a superclass of interval graphs; this result has been first generalized to the superclass of strongly chordal graphs [7] and then further extended to doubly chordal graphs [10]. Moreover, in [8] the authors present a linear time algorithm to solve the problem on circular arc graphs, a natural superclass of interval graphs. Finally, linear time algorithms for the LCST problem on cographs and co-bipartite graphs are provided in [10]. For a visual summary of the known results, see Figure 1.

Cai [4] left as an open problem to understand the complexity of the LCST problem on maximal planar graphs. In this paper we consider the subclass of maximal planar chordal graphs and give a simple linear time algorithm to determine an LCST, if it exists. Notice that this class of graphs is an interesting and well studied class, since it is equivalent to Apollonian networks [1], and to SC 3-trees [9]. Moreover, we deal also with the related class of maximal outerplanar graphs, that are equivalent to SC 2-trees [9].

In view of the equivalence of maximal planar chordal and maximal outerplanar graphs with SC 3- and SC 2-trees, respectively, and since these two are subclasses of partial k-trees, we apply to them results by Courcelle [14] and Borie *et al.* [15] to deduce that the LCST problem can be decided in linear time. Nevertheless, it is not immediate to deduce an explicit algorithm finding an LCST. So, we exploit the strong structure of maximal outerplanar and of maximal planar chordal graphs to deduce some properties leading us to design two simple linear time algorithms for finding an LCST on these classes.

The interest of this result lies in the importance to determine the exact boundary between graph classes for which the problem is hard and graph classes for which it can be polynomially solvable.

The rest of this paper is organized as follows: Section 2 is devoted to recall some known notions and to state some preliminary results that will be useful in the successive sections. In Section 3 we prove that the LCST problem is decidable in linear time on the class of partial k-trees, for any fixed k. Sections 4 and 5 are devoted to give simple linear time algorithms for finding an LCST on SC 2- and SC 3-trees, respectively, if it exists. Finally, Section 6 concludes the paper addressing some open problems.

2. Preliminaries

In this section we recall some notions, and state some preliminary lemmas that will be useful in the next sections of the paper.

As already mentioned, the considered classes of graphs (i.e. maximal outerplanar graphs and maximal planar chordal graphs) coincide with SC 2- and SC 3-trees graphs. We recall the definitions of k-tree, SC k-tree, and partial k-tree; in view of their inclusion in the latter class, from now on, we will use the name SC 2- and SC 3-trees respectively for maximal outerplanar and maximal planar chordal graphs.

Definition 1. *[9] Let k be a positive integer.*

- *The complete graph on k nodes is a k-tree; if G is a k-tree, and C is a chosen k-clique of G, then the graph obtained by adding a new node to G and connecting it to all nodes of C is a k-tree.*
- *If G is a k-tree with $|V(G)| > k$, and every k-clique has been chosen at most once in its construction, then G is called simple-clique (SC) k-tree.*
- *A spanning subgraph of a k-tree is a partial k-tree.*

In this paper we deal with the two classes of SC 2-trees and SC 3-trees. SC 2-trees coincide with maximal outerplanar graphs [9] while SC 3-trees coincide with Apollonian networks [1], and are in fact equivalent to the intersection class of chordal and maximal planar graphs [9]. Hence, these two classes represent interesting subclasses of both chordal and planar graphs.

Definition 2. *[5] Given a graph $G = (V, E)$ and two non-adjacent nodes u and v of V, a subset $S \subseteq V \setminus \{u, v\}$ is an* (u, v)-separator *if the removal of S from G separates u and v into distinct connected components.*

Let S be an (u, v)-separator of G. S is a *minimal (u, v)-separator* if no proper subset of S separates u from v. More generally, S is a *minimal separator* if it is a minimal (u, v)-separator, for some pair (u, v) of non adjacent nodes.

Given a set of nodes $V' \subseteq V$ of a graph G, we denote by $G[V']$ the *subgraph induced* in G by the nodes in V'.

We now give some properties of a minimal separator of an SC k-tree.

Lemma 1. *Let G be an SC k-tree, and S be a minimal separator in G, then the graph $G[V \setminus S]$ has exactly two connected components A_S and B_S and the two graphs $G[A_S \cup S]$ and $G[B_S \cup S]$ are SC k-trees.*

Proof. Proceed by induction on the number n of nodes in $G = (V, E)$. If $n = k + 1$, then G is a $(k + 1)$-clique and the claim is trivially true because no separator exists. Assume now that the claim is true for each SC k-tree with less than $n > k + 1$ nodes and let G be an SC k-tree with n nodes. In view of the recursive definition of the SC k-trees, there is a node t in G and a k-clique K such that $G' = G[V \setminus \{t\}]$ is an SC k-tree with $n - 1$ nodes and K is the set of neighbors of t in G. Note that the separators of G are all the separators of G' plus the separator K. Let S be a separator of G. Two cases can arise.

- *S is the k-clique K.* In this case let $A_S = V \setminus (S \cup \{t\})$ and $B_S = \{t\}$ and the claim follows since $G[A_S \cup S]$ is the SC k-tree G' and $G[B_S \cup S]$ is the SC k-tree given by the k-clique K.

- *S is also a separator in G'.* By the inductive hypothesis there exist two sets A'_S and B'_S satisfying the claim in G'. Note that it cannot be both $A'_S \cap K \neq \emptyset$ and $B'_S \cap K \neq \emptyset$ (otherwise S would not be a separator). W.l.o.g. assume $B'_S \cap K = \emptyset$ and consider $A_S = A'_S \cup \{t\}$ and $B_S = B'_S$. Note that A_S is a connected component in G (since A'_S is a connected component in G' and t is connected to at least a node in A'_S). Moreover $G[A_S \cup S]$ is the SC k-tree obtained connecting the node t to the k-clique K in the SC k-tree $G'[A'_S \cup S]$ and $G[B_S \cup S]$ is the SC k-tree $G'[B'_S \cup S]$.

□

From now on, fixed a minimal separator S, we will continue to call A_S and B_S the two connected components of $G \setminus S$.

In the following lemma we recall some simple properties of SC k-trees that can be proved by induction on the number of nodes in G and that have been stated either in [12] or in [11] for the more general class of k-trees.

Lemma 2. *Let G be an SC k-tree, then*

(i) G has $(k+1)$-cliques but no $(k+2)$-cliques,

(ii) every minimal separator of G is a k-clique,

(iii) G is a chordal graph,

(iv) For each k-clique K in G there exists a node t such that $K \cup \{t\}$ induces a $(k+1)$-clique in G.

Lemma 3. *Let G be an SC k-tree, then for any minimal separator S in G there are two nodes a and b, such that S is an (a,b)-separator; moreover $S \cup \{a\}$ and $S \cup \{b\}$ are $(k+1)$-cliques in G.*

Proof. Proceed by induction on the number n of nodes in G. If $n = k+1$, then G is a $(k+1)$-clique and the claim is trivially true because no separator exists. Assume now that the claim is true for each SC k-tree with less than n nodes and let G be an SC k-tree with n nodes. In view of the recursive definition of the SC k-trees, there is a node a in G and a k-clique K such

that $G' = G - \{a\}$ is an SC k-tree with $n-1$ nodes and K is the set of neighbors of a in G. Note that K is a (a,b)-minimal separator for G where b is a node in G' connected to all the nodes in K (the existence of such a node is ensured by item *(iv)* of Lemma 2). Thus the separator K of G satisfies the claim. Moreover all the others separators in G are also separators in G' thus the claim follows by inductive hypothesis. □

We now recall the following generalization of line graphs introduced in [6].

Definition 3. *The* k-line graph *of a graph* G*, in short* $\mathcal{L}_k(G)$*, is defined as a graph whose nodes are the* k*-cliques in* G*. Two distinct such nodes are adjacent in the* k*-line graph if and only if they have* $k-1$ *nodes in common in* G*.*

In the following, for a node X in $\mathcal{L}_k(G)$, with a small abuse of notation, we will denote by X also the set of nodes of G that are in the k-clique corresponding to X.

Lemma 4. *Let* G *be an SC* k*-tree, then a* k*-clique* S *in* G *is a minimal separator if and only if there exist two adjacent nodes* X_1 *and* X_2 *in* $\mathcal{L}_{k+1}(G)$ *such that* $S = X_1 \cap X_2$*.*

Proof. We prove the two implications separately.

($\Rightarrow$) Since S is a minimal separator in G, by Lemma 3 there are in G two nodes a and b such that S is an (a,b)-separator and $S \cup \{a\}$ and $S \cup \{b\}$ are $k+1$-cliques in G. Let X_1 and X_2 be the two nodes in $\mathcal{L}_{k+1}(G)$ corresponding to the $k+1$-cliques $S \cup \{a\}$ and to $S \cup \{b\}$ respectively. By definition of $\mathcal{L}_{k+1}(G)$, these nodes are adjacent and the claim follows.

($\Leftarrow$) We will show that the k-clique $S = X_1 \cap X_2$ in G is a minimal (a,b)-separator where $a = X_1 \setminus S$ and $b = X_2 \setminus S$.

Suppose, by contradiction, that a and b are connected in the subgraph G' induced by the nodes $V \setminus S$. Note that a and b not are adjacent in G (otherwise we have in G a $(k+2)$-clique induced by nodes of $X_1 \cup X_2$ against item *(i)* in Lemma 2) and let $P =< a, t_1, \ldots t_i, b >$, with $i \geq 1$, be a shortest path from a to b in G'. We will prove that each node of P must be connected to all the nodes in the set S, so leading to a contradiction since set $\{t_1\} \cup X_1$ forms a $(k+2)$-clique in G.

To prove that each node t_i in P must be adjacent to all the nodes in the set S, let us assume, by contradiction that there is a node t_j in P and a node u in S such that t_i and u are not adjacent. Let t be the first node

adjacent to u we meet along the path P from t_j to a and let t' be the first node adjacent to u we meet along the path P from t_i to b. Now consider in G the cycle consisting of the nodes in P from t to t' and the node u. This cycle contains at least 4 nodes (i.e. t, t_j, t' and u) and is cordless (since P is a minimal path from a to b). Thus we have a contradiction in view of item *(iii)* in Lemma 2. □

The k-line graphs have been used to obtain the following characterization of SC k-trees:

Theorem 1. *[9] A k-tree G is an SC k-tree if and only if the $(k+1)$-line graph $\mathcal{L}_{k+1}(G)$ of G is a tree.*

An SC k-tree whose $(k+1)$-line graph $\mathcal{L}_{k+1}(G)$ is a path is called *k-path.*

Since our algorithms exploit the $(k+1)$-line graph, we are interested to output $\mathcal{L}_{k+1}(G)$ in linear time from an SC k-tree input graph G; this is possible, as shown by the following result.

Lemma 5. *Let G be an SC k-tree; then tree $\mathcal{L}_{k+1}(G)$ can be computed in linear time.*

PROOF. A *perfect elimination ordering (peo)* of G is an order $v_1, v_2 \dots v_n$ of its nodes such that the set $Pred(v_i)$, $1 \leq i \leq n$, of the nodes that are adjacent to v_i in G and that precede v_i in the order, form a clique.
Rose and al. in [13] developed a method based on Lexicographic Breadth First Search (*lex*-BFS that produces a peo for chordal graphs (and obviously for SC k-trees) in linear time. Moreover in [12] Rose proved that the peo produced with this method for k-tree (and obviously for SC k-trees) has the property that the first $k+1$ nodes in the order form a $k+1$-clique and $|Pred(v_i)| = k$ for each v_i, $k+1 < i \leq n$. Once produced a peo $v_1, v_2 \dots v_n$ with these properties it is easy to construct in linear time the tree $\mathcal{L}_{k+1}(G)$. We start with a node X_1 containing the nodes $x_1, x_2, \dots x_{k+1}$. Moreover we obtain the other nodes of $\mathcal{L}_{k+1}(G)$ starting from the nodes v_i, $k+1 < i \leq n$. More precisely for v_i we create a node X_i containing the nodes $\{v_i\} \cup Pred(v_j)$, where v_j is the last predecessor of v_i, and we connect this new node X_i to the node X_j if $j > k+1$, to X_1 otherwise. It is easy to see that each of the $n-k$ nodes of the resulting tree is a $k+1$-clique and that, for each edge X_iX_j of the tree ,it holds $|X_i \cap X_j| = k$ (i.e. the tree is the graph $\mathcal{L}_{k+1}(G)$). □

Given a graph G, and one of its spanning trees T, for each node v of G, $N_T(v)$ represents the set of the nodes of G that are adjacent to v in T; these nodes will be called *T-neighbors of v*. The next lemma states a necessary condition that an LCST of a SC k-tree satisfies. As we will see later, Theorem 5 states the sufficiency of this condition for the case $k = 3$.

Lemma 6. *Let G be an SC k-tree, $k \geq 2$, S be one of its minimal separators and T be an LCST in G. We have that:*

(i) if $T[S]$ contains an isolated node, then its T-neighbors completely lie either in A_S or in B_S.

(ii) $G[S]$ contains at least one edge of T.

PROOF. Let x be an isolated node in $T[S]$. By contradiction, assume that $N_T(x)$ has a non empty intersection both with A_S and with B_S. Let a and b be the T-neighbors of x such that $a \in A_S$ and $b \in B_S$; since T is an LCST, a and b must be connected in G by either an edge or a path not passing through any other node of S, and this is a contradiction since S is a separator.

In order to prove the second assertion, assume by contradiction that subgraph $G[S]$ does not contain any edge of T. By the first assertion, each node in S has its T-neighbors either all in A_S or all in B_S. But in this case, for each node $a \in A_S$ and $b \in B_S$ it cannot exists a path in T connecting a to b. Thus T is not a spanning tree. □

3. Expressibility in monadic second order logic

In addition to the results in [2] (showing that many NP-hard problems can be linearly solved when restricted to partial k-trees), in this section we prove that also the LCST problem can be decided in linear time on partial k-trees, for every fixed k. This is a consequence of a milestone result independently discovered by Courcelle [14] and Borie *et al.* [15].

The *monadic second order logic (MSOL) of graphs* is a powerful logic in which several graph properties can be expressed. See [14] for more details. Its syntax includes:

- all the propositional connectives $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$;
- variables for nodes $v_1, v_2, ...$, edges $e_1, e_2, ...$, sets of nodes $V_1, V_2, ...$, and sets of edges $E_1, E_2, ...$;

- the quantifiers $\forall$ and $\exists$ applicable to variables;
- a binary relation $\text{inc}(e_1, v_1)$, stating that edge e_1 is incident to node v_1;
- the binary relations $v_1 \in V_1$, and $e_1 \in E_1$, for appropriate variables v_1, V_1, e_1, E_1;
- binary relations for equality, $=$, for the four types of variables.

Theorem 2. *[14] [15] Any property of graphs expressible in MSOL can be decided in linear time on partial k-trees, for any fixed k.*

The next theorem shows that the property of containing an LCST can be expressed in MSOL.

Theorem 3. *Let G be a graph. There exists an MSOL formula ϕ such that G satisfies ϕ if and only if G contains an LCST.*

PROOF. We will denote with $G[E_1]$ the subgraph induced in G by the set $E_1 \subseteq E(G)$. The proof follows the approach of [16], by first defining a statement $SpanningTree(E_1)$, expressing the fact that E_1 is the set of edges of a spanning tree of G.

We observe that operators $\cup$ and $\cap$ can be easily expressed with the rules of monadic second order logic so, in the following, in order not to overburden the exposition, we use directly them instead of their expression.

We need to define the following basic expressions:

- $v_1v_2 \in E_1 := \neg(v_1 = v_2) \wedge (\exists e_1 \in E_1)(\text{inc}(e_1, v_1) \wedge \text{inc}(e_1, v_2))$;
- $V_1 = N(v_2, E_1) := (\forall v_1)(v_1 \in V_1 \Leftrightarrow v_1v_2 \in E_1)$,
  stating that V_1 is the set of nodes adjacent to v_2 through an edge in E_1;
- $E_1 = \text{Ind}(V_1) := (\forall e_1)(e_1 \in E_1 \Leftrightarrow (\exists v_1, v_2 \in V_1)(\neg(v_1 = v_2) \wedge \text{inc}(e_1, v_1) \wedge \text{inc}(e_1, v_2)))$,
  stating that E_1 is the set of edges of the subgraph induced by V_1.
- $\text{Part}(V_1, V_2, V_3) := V_2 \neq \emptyset \wedge V_3 \neq \emptyset \wedge (V_1 = V_2 \cup V_3) \wedge (V_2 \cap V_3 = \emptyset)$,
  V_1 is partitioned by V_2 and V_3;
- $V_1 = \text{Inc}(E_1) := (\forall v_1)(v_1 \in V_1 \Leftrightarrow (\exists e_1 \in E_1)(\text{inc}(e_1, v_1)))$,
  V_1 is the set of nodes incident to edges in E_1;

- $E_1 = \mathrm{Inc}(v_1) := (\forall e_1)(e_1 \in E_1 \Leftrightarrow \mathrm{inc}(e_1, v_1))$,
  E_1 is the set of edges incident in v_1.

Given the above, we can state the connectedness of the subgraph induced by a set of edges E_1 by saying that however a bipartition of its nodes is considered, there always is an edge crossing it:

$$\mathrm{Conn}(E_1) := (\forall V_1, V_2)(\mathrm{Part}(\mathrm{Inc}(E_1), V_1, V_2) \Rightarrow (\exists v_1 \in V_1, v_2 \in V_2)(v_1 v_2 \in E_1))$$

The subgraph induced by a set of edges E_1 is 2-connected by definition if it has at least three nodes, and whenever a node is removed, the remaining graph is still connected:

$$\begin{aligned}\mathrm{Biconn}(E_1) := {} & (\forall v_1)(\mathrm{Conn}(E_1 \setminus \mathrm{Inc}(v_1))) \\ & \wedge (\exists v_1, v_2, v_3 \in \mathrm{Inc}(E_1))(\bigwedge_{1 \le i < j \le 3} v_i \neq v_j)\end{aligned}$$

The above allows us to express acyclicity of the subgraph induced by E_1 by negating the 2-connectedness of any subset of its nodes:

$$\mathrm{Acyclic}(E_1) := (\forall V_1 \subseteq \mathrm{Inc}(E_1))(\neg \mathrm{Biconn}(\mathrm{Ind}(V_1) \cap E_1))$$

And finally we can state that $G[E_1]$ is spanning tree of G with:

$$\mathrm{SpanningTree}(E_1) := \mathrm{Conn}(E_1) \wedge \mathrm{Acyclic}(E_1) \wedge (\forall v_1)(v_1 \in \mathrm{Inc}(E_1))$$

Hence, the fact that E_1 induces a locally connected spanning tree in G can be expressed as by definition:

$$\mathrm{LCST}(E_1) := \mathrm{SpanningTree}(E_1) \wedge (\forall v_1)(\mathrm{Conn}(\mathrm{Ind}(N(v_1, E_1))))$$

and the decisional problem is expressed simply as $(\exists E_1)(\mathrm{LCST}(E_1))$. □

The following corollary is an immediate consequence of Theorems 2 and 3.

Corollary 1. *The LCST problem can be decided in linear time on the class of partial k-trees, for any fixed k.*

However, as deep and interesting as it is, the result by Courcelle and Borie *et al.* does not give any immediate intuition on why and how the structure of these graphs can be exploited to solve the problem at hand. Indeed, it is a useful tool to preliminarily get a feeling of the complexity of a problem, but we feel that an explicit algorithm may provide more insight to the reader not familiar with MSOL and its relationship with partial k-trees.

4. An Algorithm to Determine an LCST of a maximal outerplanar graph

Cai [3] proved that a nontrivial graph contains an LCST if and only if it contains a spanning 2-tree T such that T does not contain as induced subgraph a 3-*sun* (shown in Figure 2). Thus, we have the following characterization.

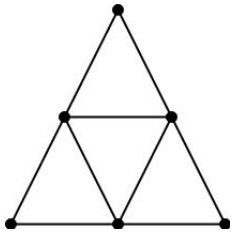

Figure 2: The 3-sun graph.

Corollary 2. *An SC 2-tree G contains an LCST if and only if G does not contain as induced subgraph a* 3-sun.

Now we prove the following result:

Lemma 7. *Let G be an SC 2-tree. Its 3-line graph $\mathcal{L}_3(G)$ has nodes of degree* 3 *if and only if G contains a 3-sun as induced subgraph.*

Proof. If G contains a 3-sun, then there are three 3-cliques all having an edge in common with the same (central) 3-clique, hence $\mathcal{L}_3(G)$ has a node of degree at least 3. Vice-versa let $X = \{x, y, z\}$ be a node of degree 3 in $\mathcal{L}_3(G)$. Let $Y_1 = \{x, y, a\}$, $Y_2 = \{x, z, b\}$ and $Y_3 = \{y, z, c\}$ be the three neighbors of X, then the six nodes $\{x, y, z, a, b, c\}$ induce a 3-sun in G. □

Now, since an SC k-tree G is a k-path if and only if $\mathcal{L}_{k+1}(G)$ is a path, from the above two results we have:

Corollary 3. *An SC 2-tree G contains an LCST if and only if $\mathcal{L}_3(G)$ is a path.*

Now we give a characterization of LCSTs of 2-paths. This characterization allows us to design an algorithm that finds an LCST of a 2-path in linear time.

Lemma 8. *Let G be a 2-path, and T be one of its spanning trees. T is an LCST if and only if, for each minimal separator $S = \{x, y\}$ of G the edge xy is in T.*

PROOF. We prove the two implications separately.

($\Rightarrow$) If T is an LCST, the claim immediately follows from property (ii) in Lemma 6.

($\Leftarrow$) Proceed by induction on the number n of nodes in G. If $n = 3$ the claim trivially holds, because no separator exists. If $n > 3$ there exists at least a separator $S = \{x, y\}$. In view of Lemma 1 and by definition of k-paths, $G[A_S \cup S]$ and $G[B_S \cup S]$ are 2-paths and hence they satisfy the inductive hypothesis, implying that $T[A_S \cup S]$ and $T[B_S \cup S]$ are LCSTs. Merging together $T[A_S \cup S]$ and $T[B_S \cup S]$ we get a tree T that is a LCST because edge xy belongs to T. □

From the above results it is easy to obtain a linear time algorithm (Algorithm 1 below) that, given an n node SC 2-tree G, returns an LCST of G if it exists, returns 'no' otherwise.

Theorem 4. *(Correctness and Complexity) Algorithm 1 determines an LCST of a given SC 2-tree if and only if it exists and runs in linear time.*

PROOF. If $\mathcal{L}_3(G)$ is not a path the algorithm, in agreement with Lemma 3, correctly returns "no".

It remains to show that the tree T constructed by the algorithm visiting the path $\mathcal{L}_3(G)$ is an LCST of G. This easily follows noting that the algorithm constructs the LCST T exploiting the characterization in Lemma 8 and selects all edges induced by each minimal separator of G. Note that, after the selection in T of the $n - 2$ minimal separators of G, it remains to connect to T the only two nodes of degree 2, z_1 and z_{n-2}, that occur in G. The node z_1 is connected in T to a node of the minimal separator $X_1 \cap X_2$ while the node z_{n-2} is connected in T to a node of of the minimal separator $X_{n-3} \cap X_{n-2}$. It is easy to see that the resulting spanning tree of G is an LCST.

For what concerns the time complexity, observe that $\mathcal{L}_3(G)$ can be computed in linear time (cf. Lemma 5) and the same asymptotic time is sufficient also to traverse the $n-2$ nodes of path $\mathcal{L}_3(G)$ to gather the edges of T. □

Algorithm 1: Algorithm FindLCSTinSC2trees

Input: an n node SC 2-tree G
Output: an LCST of G if it exists, NO otherwise
Compute tree $\mathcal{L}_3(G)$;
if $\mathcal{L}_3(G)$ *is not a path* **then**
  return NO;
end
Let $X_1, X_2, \ldots X_{n-2}$ be a linear order of the nodes of the path $\mathcal{L}_3(G)$;
$T \leftarrow \emptyset$;
if $\mathcal{L}_3(G)$ *consists of a single node* X_1 **then**
  insert in T any two edges of the 3-clique X_1;
  return T;
end
for $i = 1$ *to* $n-3$ **do**
  add to T the edge connecting the two nodes in the minimal separator $X_i \cap X_{i+1}$;
end
Let $X_j = \{x_j, y_j, z_j\}$ for $j \in \{1, n-2\}$;
Let x_1y_1 be the edge in T for $X_1 \cap X_2$ and $x_{n-3}y_{n-3}$ be the edge in T for $X_{n-3} \cap X_{n-2}$;
Add to T the two edges z_1x_1 and $z_{n-2}x_{n-2}$;
return T.

5. An Algorithm to Determine an LCST of a maximal planar chordal graph

In the previous section, we have seen that it is easy to determine an LCST of an SC 2-tree G, if it exists, exploiting its $\mathcal{L}_3(G)$. Unfortunately, when we move to SC 3-trees, things seem to be not so easy anymore. Nevertheless, we will show that it is possible to determine an LCST of an SC 3-tree G, if it exists, exploiting its $\mathcal{L}_4(G)$, in linear time. This is the aim of this section.

In the following, the graph $2K_2$ is the disjoint union of two copies of K_2 (where K_2 is the complete graph on two nodes).

Lemma 9. *Let G be an SC 3-tree and T be one of its spanning trees. If, for each minimal separator $S = \{x, y, z\}$ of G, one of the following is true:*

(i) T contains exactly two edges of $G[S]$;

(ii) T contains exactly one edge of $G[S]$ (w.l.o.g. this edge is xy) and either $N_T(z) \subseteq A_S$ or $N_T(z) \subseteq B_S$;

then, for each node X in $\mathcal{L}_4(G)$ it holds $T[X] \neq 2K_2$.

PROOF. Suppose, by contradiction, that there exists a 4-clique $X = \{x, y, z, t\}$ in $G = (V, E)$ such that $T[X] = 2K_2$. This implies that, for each minimal separator $S \subset X$ of G, it holds (ii). Let $S = \{x, y, z\}$ be any of these separators and let z be the isolated node in $T[S]$. Without loss of generality let B_S be the component of $G[V \setminus S]$ containing $N_T(z)$. In graph $T[A_S \cup S]$ node z is hence isolated. The above reasoning applies to any other separator in X. Thus the path in T connecting the two edges of the $2K_2$ must be in X and this contradicts the assumption that $T[X] = 2K_2$. □

Lemma 10. *Let G be an SC 3-tree and T be one of its spanning trees. If, for each minimal separator $S = \{x, y, z\}$ of G, the following is true:*

(ii) T contains exactly one edge of $G[S]$ (w.l.o.g. this edge is xy) and either $N_T(z) \subseteq A_S$ or $N_T(z) \subseteq B_S$;

then G is a 3-path and T is an LCST.

PROOF. First we show that G is a 3-path. By contradiction assume that tree $\mathcal{L}_4(G)$ is not a path. Let $X = \{x, y, z, t\}$ be a node of $\mathcal{L}_4(G)$ with three neighbors Y_1, Y_2 and Y_3. Consider now the three minimal separators $S_i = X \cap Y_i$, $1 \leq i \leq 3$ (cf. Lemma 4). In order to fix the ideas let $t \in X$ be the node in $\bigcap_i Y_i$ and $S_1 = \{x, y, t\}$, $S_2 = \{x, z, t\}$ and $S_3 = \{z, y, t\}$. In each graph $T[S_i]$, $1 \leq i \leq 3$, there is a single isolated node, if this node is t for all the three graphs, then xy, xz and yz are in T. Thus the set $\{x, y, z\}$ is a cycle in T, a contradiction. Hence there is a minimal separator S_i,$1 \leq i \leq 3$, such that t is not isolated in $T[S_i]$, w.l.o.g. let it be S_1 and let $xt \in E(T)$. This implies that y is the only isolated node in $T[S_1]$, z is the only isolated node in $T[S_2]$ and the edges xy, ty, xz and tz are not in T. This in turn implies that yz is the only edge in $T[S_3]$. Summarizing, we have that the edges xt and yz form a $2K$ in the 4-clique X, a contradiction to Lemma 9.

Now we show that T is an LCST. We proceed by induction on the number n of nodes in the 3-path $G = (V, E)$. If $n = 4$, then G is a 4-clique and

the claim is trivially true because any spanning tree of a 4-clique is locally connected. Assume now that the claim is true for every 3-path with less than $n > 4$ nodes and let G be a 3-path with n nodes. Let t be the last node added to G in its recursive definition and consider the separator $S = \{x, y, z\}$ identified by the neighbors of t in G. Observe that $G' = G[V \setminus \{t\}]$ is a 3-path with $n - 1$ nodes since $\mathcal{L}_4(G')$ can be obtained by $\mathcal{L}_4(G)$ by cutting the leaf containing t. Moreover, for all the separators of G' (note that these separators are also separators of G) $T' = T[V \setminus \{t\}]$ satisfies (ii) in G'. Since T satisfies (ii) for the separator S in G, t cannot be adjacent in T to both x and y (otherwise a cycle is introduced in a tree), so the degree of t in T is at most 2. We will examine the two cases.

1. $|N_T(t)| = 1$: T' is a spanning tree for G' and, by inductive hypothesis, it is an LCST of G'. Moreover, since T is connected it must be either $xt \in T$ or $yt \in T$ (remember that S satisfies (ii)). W.l.o.g. assume that $xt \in T$. For each $u \in V$, it holds

$$N_T(u) = \begin{cases} \{x\} & \text{if } u = t \\ N_{T'}(x) \cup \{t\} & \text{if } u = x \\ N_{T'}(u) & \text{otherwise} \end{cases}$$

   Now, using the fact that T' is an LNCS for G' and that $yt \in E$ and $y \in N_{T'}(x)$, it is easy to see that T is an LCST for G.
2. $|N_T(t)| = 2$: Without loss of generality assume that $N_T(t) = \{x, z\}$. Note that $N_T(z) = \{t\}$ since S satisfies (ii) in T. Let a be a node in G connected to all nodes in S (such a node there exists by Lemma 3). Since G is a 3-path, any induced 4-clique (in particular, $K = \{a, x, y, z\}$) contains at most 2 minimal separators, so besides $\{x, y, z\}$, at most one among $\{x, y, a\}$, $\{x, z, a\}$ and $\{y, z, a\}$ is a minimal separator contained in K. There are three cases to consider.
   (a) *S is the only minimal separator in K.* In this case G has only five nodes (the nodes $\{a, x, y, z, t\}$). The node a in T can be connected only to x or to y.
   In the first case (where ax is in T), we have $N_T(x) = \{y, t, a\}$ and $N_T(t) = \{x, z\}$. Moreover x and t are the only nodes in T having degree greater than one. Thus we conclude that T is an LCST by noting that ay, yt and xz are edges of G.
   In the second case (where ay is in T), x, t and y are the only nodes in T having degree greater then one and $N_T(x) = \{y, t\}$, $N_T(t) = \{x, z\}$ and $N_T(y) = \{x, a\}$. Thus again T is an LCST by noting that yt, xz and xa are edges of G.

(b) $S' = \{a, x, y\}$ *is a minimal separator in* K. $A_{S'}$ contains only t and z and $T' = T[B_{S'} \cup S']$ is a spanning tree in $G' = G[B_{S'} \cup S']$ that satisfies (ii) on every minimal separator, hence -by inductive hypothesis - G' is a 3-path and T' is an LCST. Summarizing, for each $u \in V$, it holds

$$N_T(u) = \begin{cases} \{z, x\} & \text{if } u = t \\ \{t\} & \text{if } u = z \\ N_{T'}(x) \cup \{t\} & \text{if } u = x \\ N_{T'}(u) & \text{otherwise} \end{cases}$$

Now, using that T' is an LCST of G', $zx \in E$, $y \in N_{T'}(x)$ and $yt \in E$, it is easy to see that T is a LCST of G.

(c) S' *is a minimal separator in* K *and either* $S' = \{a, y, z\}$ *or* $S' = \{a, x, z\}$. Assume first that $S' = \{a, y, z\}$. Note that T' obtained by adding edge xz to $T[V \setminus \{t\}]$ is a spanning tree of $G' = G[V \setminus \{t\}]$ and it satisfies (ii). Thus, by inductive hypothesis, T' is an LCST of G'. Note that S' satisfies (ii) on T and since $N_T(z) = \{t\}$ it must be $ay \in E(T)$. Summarizing, for each $u \in V$, it holds

$$N_T(u) = \begin{cases} \{z, x\} & \text{if } u = t \\ \{t\} & \text{if } u = z \\ \{y, t\} & \text{if } u = x \\ N_{T'}(u) & \text{otherwise} \end{cases}$$

Now, using that T' is an LCST of G', and zx and yt are edges of G, it is easy to see that T is an LCST of G.
The reasoning is similar if we assume that the minimal separator S' is $\{a, x, z\}$. In this case we can consider the spanning tree T' of G' obtained by adding the edge yz to $T[V \setminus \{t\}]$. □

We now prove a characterization that will allow us to design a linear time algorithm to determine an LCST of an SC 3-tree.

Theorem 5. *Let* G *be an SC* 3*-tree, and* T *be one of its spanning trees.* T *is an LCST if and only if, for each minimal separator* $S = \{x, y, z\}$*, one of the following is true:*

(i) T *contains exactly two edges of* $G[S]$*;*

(ii) T *contains exactly one edge of* $G[S]$ *(w.l.o.g. this edge is* xy*) and either* $N_T(z) \subseteq A_S$ *or* $N_T(z) \subseteq B_S$*.*

PROOF. We prove the two implications separately.

($\Rightarrow$) Let T be an LCST, and let us prove that either *(i)* or *(ii)* hold on S.

First, notice that from item *(ii)* of Lemma 2, $G[S]$ is a 3-clique, so it cannot contain three edges of T, otherwise a cycle would occur in T; so, in view of Lemma 6, $G[S]$ contains either one or two edges of T. If it contains exactly two edges of T, then *(i)* holds and we have done. If, on the contrary, $G[S]$ contains exactly one edge xy of T then, by Lemma 6, $N_T(z)$ has an empty intersection either with A_S or with B_S, that is *(ii)* holds.

($\Leftarrow$) Let us now assume that S satisfies either *(i)* or *(ii)*, and let us prove that T is an LCST. The proof proceeds by induction on the number n of nodes of G. For $n = 4$ (the basis of the induction) G is a 4-clique and each spanning tree of G is an LCST and the claim is trivially true since no separator exists. Assume now that G has $n > 4$ nodes and the claim is true for every SC 3-tree with less than n nodes. If all the minimal separators of G satisfy (ii), by Lemma 10 we have that G is a 3-path and T is an LCST.

It remains to consider the case in which there exists a separator $\tilde{S}$ of G that satisfies *(i)*. Consider graphs $G_1 = G[A_{\tilde{S}} \cup \tilde{S}]$ and $G_2 = G[B_{\tilde{S}} \cup \tilde{S}]$ and the spanning trees $T_1 = T[A_{\tilde{S}} \cup \tilde{S}]$ of G_1 and $T_2 = T[B_{\tilde{S}} \cup \tilde{S}]$ of G_2. In view of Lemma 1, graphs G_1 and G_2 are SC 3-trees and each separator of one of these two graphs is in fact a separator of G hence, for each separator S of G_i, $1 \leq i \leq 2$, the tree T_i satisfies either *(i)* or *(ii)*. By inductive hypothesis, it follows that T_1 and T_2 are LCSTs of G_1 and G_2, respectively. Moreover, we have

$$N_T(u) = \begin{cases} N_{T_1}(u) & \text{if } u \in A_{\tilde{S}} \\ N_{T_2}(u) & \text{if } u \in B_{\tilde{S}} \\ N_{T_1}(u) \cup N_{T_2}(u) & \text{if } u \in \tilde{S} \end{cases}$$

For each u not in $\tilde{S}$, we already know that its T-neighbors are connected in $G \setminus \{u\}$ (since T_1 is an LCST of G_1 and T_2 in an LCSTs of G_2); for each u in $\tilde{S}$, its T-neighbors are partially in $A_{\tilde{S}}$ (and they are connected in $A_{\tilde{S}} \cup \tilde{S}$), partially in $A_{\tilde{S}}$ (and they are connected in $B_{\tilde{S}} \cup \tilde{S}$), and partially in $\tilde{S}$ (through which all the T-neighbors of u are connected since $\tilde{S}$ is a 3-clique in G). It follows that T is an LCST of G. □

Definition 4. *Let $S = \{x, y, z\}$ be a minimal separator of an SC 3-tree G. Let G' be the graph obtained from G by substituting set B_S with the single node $\{b\}$ connected to all the three nodes in S. A* partial solution on $A_S \cup S$ w.r.t. *S, $H_{A_S \cup S}$, is a spanning forest of $A_S \cup S$ such that there exists an LCST T of G' such that $H_{A_s \cup S} = T[A_S \cup S]$.*

In the following we will call simply H a partial solution when S and $A_S \cup S$ are clear from the context.

Theorem 5, characterizing LCSTs in SC 3-trees, suggests that partial solutions fall in exactly three distinct categories, depending on how many edges of H are induced in S, and depending on the presence or not of an isolated node in H. Next definition formalizes this fact.

Definition 5. *Let $S = \{x, y, z\}$ be a minimal separator of an SC 3-tree G. Let H be a partial solution on $A_S \cup S$ w.r.t. S. We say that H has label:*

- *α_x if yz is an edge of H and x is isolated in H;*
- *β_x if xy and xz are both in H;*
- *γ_x if yz is an edge of H, xy and xz are not in H, and x is not isolated in H.*

Analogous definitions can be given for labels α_y, β_y and γ_y, and α_z, β_z and γ_z.

Definition 6. *Let S be a minimal separator in G, and assume $|A_S| = 1$. The canonical partial solutions of $G[A_S \cup S]$ associated to label χ_v (with $\chi \in \{\alpha, \beta, \gamma\}$ and $v \in S$) are depicted below.*

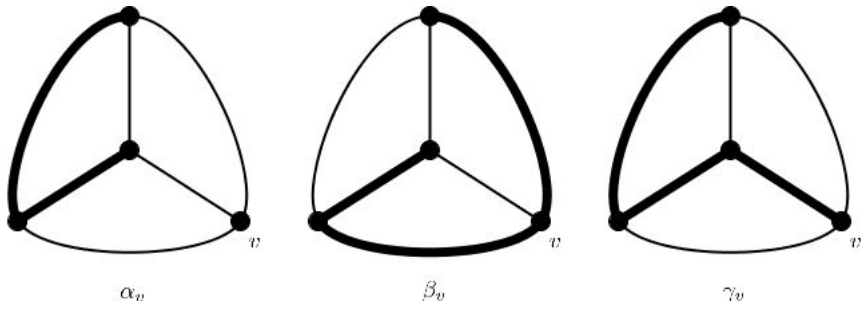

Figure 3: The three canonical partial solutions. The three outer nodes are in S, while the central node is in A_S.

We are now ready to describe the algorithm that, given an SC 3-tree, determines an LCST if it has one. We highlight that, in order not to overburden the exposition, we focus on the decisional problem. It is not difficult, given the information gathered in the decisional version of the algorithm, to find the edges of the LCST, as will be explained later.

From now on, we assume $\mathcal{L}_4(G)$ to be rooted in a degree 1 node, R. For any node $X \neq R$, we denote by $f(X)$ its parent, that is the first node encountered on the unique path from X to R; by $\bar{X}$ we denote the set of nodes of $\mathcal{L}_4(G)$ in the subtree rooted at X. To not clutter the exposition we will sometimes denote with $\bar{X}$ also the corresponding set of nodes in G, that is $\{x \in V(G) : x \in Y \text{ and } Y \in \bar{X}\}$.

Since we established the equivalence between minimal separators of G and edges of $\mathcal{L}_4(G)$ (cf. Lemma 4), in the following we will identify a minimal separator $S = X \cap f(X)$ of G with the corresponding edge $Xf(X)$ of $\mathcal{L}_4(G)$; moreover, we define the set of labels of edge $Xf(X)$ as $L(Xf(X)) = \{\chi_v : \exists \text{ a partial solution on } \overline{X} \text{ w.r.t. } X \cap f(X) \text{ with label } \chi_v\}$.

The very high level idea of the algorithm consists in traversing $\mathcal{L}_4(G)$ in post-order; when visiting a node X, we compute the set $L(Xf(X))$ of labels using the sets of labels of the children of X, $Y_1, ..., Y_c$, which have already been computed. This is done with the aim of extending the partial solutions of $\overline{Y_i}$, combining them in a partial solution of $\overline{X}$. It is clear that G contains an LCST if and only if $L(YR) \neq \emptyset$, where Y is the only child of root R.

Algorithm 2: Algorithm Compute-Labels

Input: An edge $Xf(X)$ of $\mathcal{L}_4(G)$
Output: The set of labels $L(Xf(X))$
$G' \leftarrow G[X \cup f(X) \cup \bigcup_{i=1}^{c} Y_i]$;
$L(Xf(X)) \leftarrow \emptyset$;
for *each* $(\chi^1_{v_1}, ..., \chi^c_{v_c}) \in L(Y_1X) \times ... \times L(Y_cX)$ **do**
  Let H_i be the canonical partial solution of $G'[Y_i]$ associated to $\chi^i_{v_i}$, $i = 1, ..., c$;
  for *each subset* $E' \subseteq E(X)$ **do**
    $H \leftarrow (X \cup \bigcup_{i=1}^{c} Y_i, E' \cup \bigcup_{i=1}^{c} E(H_i))$;
    if *H is a partial solution of $G'[X \cup \bigcup_{i=1}^{c} Y_i]$ w.r.t. separator $Xf(X)$* **then**
      Add to $L(Xf(X))$ the label corresponding to H;
    end
  end
end
return $L(Xf(X))$.

We now focus on the issue of assigning to an edge $Xf(X)$ its set of labels $L(Xf(X))$. First of all, notice that if X is a leaf, then the partial solutions of $\overline{X}$ are exactly the nine canonical partial solutions, so in this case $L(Xf(X))$

contains all nine labels, that is $L(Xf(X)) = \{\chi_v | \chi \in \{\alpha, \beta, \gamma\}, v \in X \cap f(X)\}$. Otherwise, assume for example that X has two children Y_1, and Y_2. By brute force we test every pair of 2 labels, each one from the set $L(Y_1X) \times L(Y_2X)$, that is the cartesian product of $L(Y_1X)$ and $L(Y_2X)$. We "decode" these labels in the corresponding canonical partial solution, which we combine, together with a subset of edges E' of $E(X)$, in a subgraph H of $G[X \cup Y_1 \cup Y_2]$. If this subgraph is a partial solution of $G[X \cup Y_1 \cup Y_2 \cup f(X)]$ w.r.t. separator $X \cap f(X)$, then we add the corresponding label to $L(Xf(X))$; the following algorithm does the job, and its correctness is proved below.

Notice that, when $c = 0$, the cartesian product $L(Y_1X) \times ... \times L(Y_cX)$ is by definition equal to the set containing the empty tuple $\{()\}$; so, when X is a leaf, the external cycle is executed exactly once, the H_i's do not exist, and $H = (X, E')$ in every iteration of the inner cycle. Also, notice that when at least one of the $L(Y_iX)$'s is empty, then $L(Y_1X) \times ... \times L(Y_cX) = \emptyset$, so the outer cycle is never executed, and the output $L(Xf(X))$ is the empty set.

Before proving the correctness of Algorithm 2, we highlight that its time complexity is constant, since the cardinality of $L(Y_1X) \times ... \times L(Y_cX)$ is always at most 9^3, there are a constant number of subsets of $E(X)$, and the "if" condition can be verified in constant time since G' has size $O(c) = O(1)$. Notice that Algorithm 2 may be substituted by a constant size look-up table, if one is interested in the overall efficiency of the algorithm (see Figure 5 in Appendix).

The following lemma is needed in the proof of the correctness of algorithm 2. Since it is an immediate consequence of the definition of partial solution, its proof is omitted.

Lemma 11. *Let G be a SC 3-tree, $Xf(X)$ be an edge of $\mathcal{L}_4(G)$. Let $G' = G[(V \setminus \overline{X}) \cup X]$. Assume G has a LCST T such that $T[\overline{X}]$ is a partial solution with label χ_v. Then, G' has a LCST T' such that $T'[X]$ is the canonical partial solution associated to label χ_v.*

Lemma 12. *After the execution of Algorithm 2 on input $Xf(X)$, $L(Xf(X))$ contains label χ_v, ($\chi \in \{\alpha, \beta, \gamma\}$ and $v \in X \cap f(X)$) if and only if there exists a partial solution of $G[\overline{X}]$ having label χ_v.*

Proof. ($\Rightarrow$) If X is a leaf, that is $c = 0$, then the canonical partial solution on $G[\overline{X}] = G[X]$ corresponding to label χ_v satisfies the statement.

Else, assuming $1 \leq c \leq 3$, let E' be the selected subset of $E(X)$ such that H is a partial solution of $G'[X \cup \bigcup_{i=1}^{c} Y_i]$ w.r.t. separator $Xf(X)$, for

which label χ_v was added to $L(Xf(X))$. By structural induction, there are partial solutions $\overline{H}_i$ of $G[\overline{Y}_i]$, having label $\chi^i_{v_i}$, $i = 1, ..., c$. By Lemma 11, if $(\overline{X}, E' \cup \bigcup_{i=1}^{c} E(\overline{H}_i))$ was not a partial solution of $G[\overline{X}]$, then neither would H be a partial solution of $G'[X \cup \bigcup_{i=1}^{c} Y_i]$, a contradiction. Finally, notice that $G[\overline{X}]$ has the same label of H, that is χ_v.

($\Leftarrow$) If X is a leaf, then $G[\overline{X}] = G[X]$ admits all nine possible canonical partial solutions. Since $c = 0$, the outer cycle is executed exactly once on the empty tuple (), and the inner cycle will find a partial solution for each possible label. Assume now that $1 \leq c \leq 3$. Assume there exists a partial solution $\overline{H}$ of $G[\overline{X}]$ with respect to separator $Xf(X)$. Then, notice that $\overline{H}[\overline{Y}_i]$ is a partial solution of $G[\overline{Y}_i]$ with respect to separator Y_iX with label $\chi^i_{v_i}$, so by structural induction $L(Y_iX)$ contains label $\chi^i_{v_i}$, $i = 1, ..., c$. Let H_i be the canonical partial solution of $G[Y_i]$ with respect to separator Y_iX. Then, by Lemma 11, $H = (X \cup \bigcup_{i=1}^{c} Y_i, E' \cup \bigcup_{i=1}^{c} E(H_i))$ is a partial solution of $G'[X \cup \bigcup_{i=1}^{c} Y_i]$ with respect to separator $Xf(X)$, where $E' = E(\overline{H}[X])$ is found by brute force by the inner cycle, and the corresponding label (that is the same as $\overline{H}$) is added to $L(Xf(X))$. □

We are ready to give the pseudocode of the algorithm deciding whether an SC 3-tree has an LCST or not.

Algorithm 3: Algorithm Decide-LCSTonSC3-trees

```
Input: An n node SC 3-tree G
Output: YES if an LCST of G exists, NO otherwise
if n = 4 then
    return YES;
end
Compute L_4(G);
Root L_4(G) in a degree 1 node R;
Let Y be the only child of R;
for each (χ^1_{v_1}, ..., χ^c_{v_c}) ∈ L(Y_1X) × ... × L(Y_cX) do
    L(Xf(X)) ← Compute-LabelsXf(X);
    if L(YR) ≠ ∅ then
        return NO;
    else
        return YES;
    end
end
```

Theorem 6. *(Correctness and Complexity) Algorithm 3 returns "yes" if and only if the SC 3-tree in input has an LCST, in linear time.*

PROOF. If $n = 4$, then obviously any spanning tree of G is locally connected, and the algorithm returns "yes". Otherwise, by Lemma 12, we have that $L(YR)$ is nonempty if and only if there exists a partial solution H of $G[\overline{Y}]$ with respect to the minimal separator YR. It is easy to see that H can be extended to a LCST of G adding a single edge.

Moreover, the algorithm is linear. Indeed, $\mathcal{L}_4(G)$ can be computed in linear time (cf. Lemma 5), and Algorithm 2 is called $O(n)$ times, and, as already noted, has constant cost. □

In order not to overburden the exposition, we do not detail how to reconstruct an LCST from the labels assigned in the algorithm, so here we will give only an overview. We can traverse again $\mathcal{L}_4(G)$, this time in a pre-order fashion; starting from the edge incident to the root, we arbitrarily choose one label; this label implies a certain canonical partial solution, so we add the corresponding edges to the current LCST. At the general iteration, we proceed visiting the children of the current node having already chosen a label of the separator corresponding to the edge connecting it with its father; this label came up from precise labels on the edges connecting this node to its children, so we are forced to choose exactly those labels, and we add in the LCST the corresponding edges of G.

We conclude this section by presenting a complete example showing how our algorithm works.

Example. Consider graph G in Figure 4.a, whose rooted $\mathcal{L}_4(G)$ is depicted in Figure 4.b. Performing a post-order traversal of $\mathcal{L}_4(G)$, we first label edges $S_1 = \{2, 3, 4\}$ and $S_2 = \{2, 4, 5\}$ with all 9 labels.

When moving to label $S_3 = \{3, 4, 5\}$, we construct $L(S_3) = \{\alpha_3, \alpha_5, \beta_3, \beta_4, \beta_5, \gamma_3, \gamma_4, \gamma_5\}$. Then, we conclude labeling $S_4 = \{3, 5, 6\}$ with $L(S_4) = \{\alpha_3, \alpha_5, \alpha_6, \beta_3, \beta_5, \beta_6, \gamma_3, \gamma_5, \gamma_6\}$.

Since we labeled the only edge coming out from the root of $\mathcal{L}_4$, we deduce that G admits an LCST.

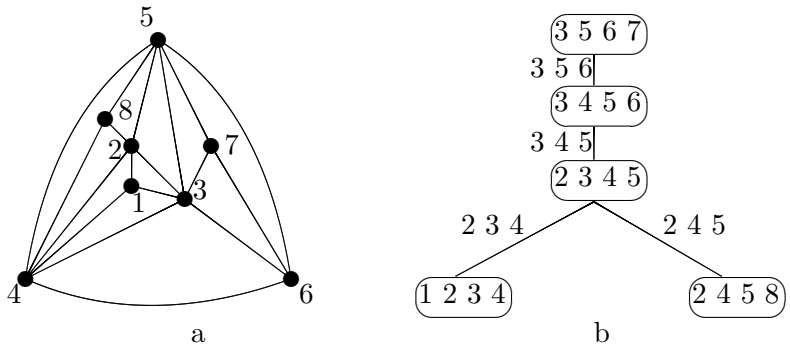

Figure 4: Example of how algorithm Decision-LCSTonSC3-trees works.

6. Conclusions and Open Problems

Many open questions arise from this paper.

In [4], Cai posed as open problem the characterization of k-trees containing a 3-sun free spanning 2-tree (equivalently, an LCST). Exploiting results by Courcelle [14] and Borie *et al.* [15], we proved that the LCST problem is decidable in linear time on the class of partial k-trees, for a fixed k, by showing that the problem is expressible in monadic second order logic of graphs. This leads to a linear time decisional procedure effectively distinguishing the partial k-trees containing an LCST from the other ones, nevertheless we consider the problem still open, as a clean graph theoretical characterization would be more satisfying and informative.

We have then given explicit linear time algorithms finding an LCST (if it exists) on the interesting subclasses of maximal outerplanar graphs (SC 2-trees), and maximal planar chordal graphs (SC 3-trees). The latter result is a first attack to the open problem left in [4] requiring to establish the complexity of the LCST problem on the wider class of maximal planar graphs.

The approach we used for SC 3-trees is clearly generalizable to SC k-trees, for $k > 3$. Of course, the complexity of these algorithms (one for each k) would be linear in n and exponential in k. Indeed, it is neither practical nor interesting to even attempt to give an explicit and detailed exposition for the case $k > 3$, since all the complexity of the problem is already contained and exemplified by the case $k = 3$. Nevertheless, we want to briefly discuss the problem in the general case.

First of all, an extension of Theorem 5 would be needed, becoming:

Let G be a SC k-tree, and T be a spanning tree of G. T is locally connected if and only if for every minimal separator S of G, and for every

$x \in S$:

$$N_T(x) \cap S = \emptyset \Rightarrow N_T(x) \subseteq A_S \text{ or } N_T(x) \subseteq B_S.$$

Accordingly, we would need to generalize the descriptions of the labels. Recall that our labels are an encoding of a partial solution H found up to a certain minimal separator S of the input graph, and devoid of all the information about H which are not needed to proceed with the computation. Explicitly, these labels should contain two pieces of information:

- $E(H[S])$, that is, the edges of the minimal separator S which the partial solution contains (notice that $|E(H[S])| > 1$, and it is the edge set of a forest);
- a partition of the set of the connected components of $E(H[S])$: in each partition set those components which are all connected together through the already computed portion of the partial solution H.

Notice that these forests necessarily contain at least one edge due to the above extension of Theorem 5: since T is a tree, and it has to pass through every separator, there must be at least one vertex v for each minimal separator S such that $N_T(v) \cap A_S$ and $N_T(v) \cap B_S$ are not empty, and since T is locally connected, $N_T(v) \cap S$ is not empty.

Also extending Algorithm 2 is conceptually easy but laborious in practice: the criteria to join the generalized labels together, and thus increasing the partial solution, are the same as for the case $k = 3$. So, for every k-tuple of labels (each from one of the at most k children) it is necessary to check (1) whether they acyclically overlap on the edges of the minimal separator they share (first item of the list above is sufficient to decide this), and (2) whether cycles are not created by their union outside the minimal separator (and this condition can be checked using the second item).

A natural question to ask is how many labels there are for a fixed k. Let F_k be the set of labeled forests on k vertices with at least one edge; let C_f be the number of connected components of a forest $f \in F_k$. From the discussion above, it follows that the number of labels needed are $\sum_{f \in F_k} B_{C_f}$, where B_n denotes the n-th Bell number, the number of possible partitions of a set of n elements.

Finally, a minor modification of this algorithm would allow the enumeration of all LCSTs of an SC k-tree, still running in polynomial time, for any fixed k. Indeed, the algorithm would have polynomial delay and the number of LCSTs in an SC k-tree is upper bounded by $nf(k)$, where f is the number of possible labels. This follows from the fact that an SC k-tree has at most n minimal separators, and that an LCST can behave in at most

$f(k)$ different ways on a minimal separator.

7. Bibliography

[1] J.S. Andrade, H.J. Herrmann, R.F.S. Andrade, L.R. da Silva, Apollonian Networks: Simultaneously Scale-Free, Small World, Euclidean, Space Filling, and with Matching Graphs *Phys. Rev. Lett.* 94, 2005. Erratum in *Phys. Rev. Lett.* 102, 2009.

[2] S. Arnborg, and A. Proskurowski, Linear Time Algorithms for NP-Hard Problems Restricted to Partial k-trees. *Discrete Applied Mathematics* 23, 11–24, 1989.

[3] L. Cai, On spanning 2-trees in a graph. *Discrete Applied Mathematics* 74, 203–216, 1997.

[4] L. Cai, The complexity of the locally connected spanning tree problem. *Discrete Applied Mathematics* 131, 63–75, 2003.

[5] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, 1980.

[6] V.B. Le, Perfect k-line graphs and k-total graphs, *J. Graph Theory* 17, 65–73, 1993.

[7] C.-C. Lin, G.J. Chang, G.-H. Chen, Locally connected spanning trees in strongly chordal graphs and proper circular-arc graphs. *Discrete Mathematics*, 307(2),208–215, 2007.

[8] C.-C. Lin, G.-H. Chen, G.J. Chang, A linear-time algorithm for finding locally connected spanning trees on circular-arc graphs. *Algorithmica*, 66(2), 369–396, 2013.

[9] L. Markenzon, C.M. Justel, and N. Paciornik, Subclasses of k-trees: Characterization and recognition. *Discrete Applied Mathematics* 154(5), 818–825, 2006.

[10] B.S. Panda and D. Pradhan, Locally connected spanning trees in cographs, complements of bipartite graphs and doubly chordal graphs. *Information Processing Letters*, 110(23), 1067–1073, 2010.

[11] A. Proskurowski, Separating Subgraphs in k-trees: Cables and Caterpillars. *Discrete Mathematics* 49, 275–285, 1984.

[12] D.J. Rose, On simple characterizations of k-trees. *Discrete Mathematics*, 7, 317–322, 1974.

[13] D.J. Rose, R.E. Tarjan, G. Luker, Algoritmic aspects of vertex elimination on graphs, *SIAM J. Comput. 5*, 266–283, 1976.

[14] B. Courcelle, J. Engelfriet, Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach, Encyclopedia of mathematics and its applications, *Cambridge University Press*, 138, 2012.

[15] R. B. Borie, P. R. Gary, C. A. Tovey, Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families, *Algorithmica*, , 7(1), 555–581, 1992.

[16] H. L. Bodlaender, F. V. Fomin, P. A. Golovach, Y. Otachi, E. J. van Leeuwen, Parameterized Complexity of the Spanning Tree Congestion Problem, *Algorithmica*, 64(1), 85–111, 2012.

Appendix

Table used in Algorithm Decide-LCSTonSC3-trees

label of $\{x,y,t\}$	label of $\{x,z,t\}$	label of $\{y,z,t\}$	label of $\{x,y,z\}$
α_x	-	-	$\alpha_x(+yz), \beta_z(+xz,yz), \gamma_y(+zt,xz)$
α_y	-	-	$\alpha_y(+xz), \beta_z(+xz,yz), \gamma_x(+yz)$
α_t	-	-	$\beta_x(+xz,zt), \beta_y(+yz,zt)$
β_x	-	-	$\alpha_z, \beta_x(+xz), \beta_y(+yz), \gamma_z(+tz)$
β_y	-	-	$\alpha_z, \beta_x(+xz), \beta_y(+yz), \gamma_z(+tz)$
β_t	-	-	$\gamma_y(+xz), \gamma_x(+yz)$
γ_x	-	-	$\gamma_y(+xz), \gamma_x(+yx)$
γ_y	-	-	$\gamma_y(+xz), \gamma_x(+yz),$
γ_t	-	-	$\alpha_z, \beta_x(+xz), \beta_y(+yz)$
α_x	α_t	-	$\beta_z(+yz)$
α_x	β_z	-	γ_y
α_t	α_x	-	$\beta_y(+yz)$
α_t	β_z	-	β_x
α_t	γ_t	-	β_x
β_x	α_z	-	$\alpha_z, \beta_y(+yz)$
β_x	γ_z	-	γ_z
β_x	β_x	-	β_x
β_x	β_t	-	γ_z
β_y	α_x	-	γ_z
β_y	α_t	-	β_x
β_t	α_z	-	$\gamma_x(+yz)$
β_t	β_x	-	γ_y
γ_y	β_x	-	γ_y
γ_t	α_t	-	β_x
α_x	α_t	β_y	β_z
α_x	β_z	β_t	γ_y
α_t	α_x	β_z	β_y
α_t	β_z	α_y	β_x
β_x	α_z	α_t	β_y
β_x	β_t	α_y	γ_z
β_y	α_x	β_t	γ_z
β_y	α_t	α_z	β_x
β_t	α_z	β_y	γ_x
β_t	β_x	α_z	γ_y

Figure 5: Table for the construction of the labels. Symbol '-' means that the corresponding child is not present in $\mathcal{L}_4$. Notice that missing combinations do not lead to any feasible label for separator $\{x,y,z\}$.