Subexponential Algorithms for Rectilinear Steiner Tree and Arborescence Problems

FEDOR V. FOMIN, University of Bergen
DANIEL LOKSHTANOV, University of California Santa Barbara
SUDESHNA KOLAY, Indian Institute of Technology Kharagpur
FAHAD PANOLAN, Indian Institute of Technology Hyderabad
SAKET SAURABH, Homi Bhabha National Institute, IRL 2000 ReLaX, and University of Bergen

A *rectilinear Steiner tree* for a set K of points in the plane is a tree that connects K using horizontal and vertical lines. In the Rectilinear Steiner Tree problem, the input is a set $K = \{z_1, z_2, \ldots, z_n\}$ of n points in the Euclidean plane ($\mathbb{R}^2$), and the goal is to find a rectilinear Steiner tree for K of smallest possible total length. A *rectilinear Steiner arborescence* for a set K of points and a root $r \in K$ is a rectilinear Steiner tree T for K such that the path in T from r to any point $z \in K$ is a shortest path. In the Rectilinear Steiner Arborescence problem, the input is a set K of n points in $\mathbb{R}^2$, and a root $r \in K$, and the task is to find a rectilinear Steiner arborescence for K, rooted at r of smallest possible total length. In this article, we design deterministic algorithms for these problems that run in $2^{O(\sqrt{n}\log n)}$ time.

CCS Concepts: • **Theory of computation → Fixed parameter tractability**;

Additional Key Words and Phrases: Rectilinear Steiner tree, rectilinear Steiner arborescence, subexponential exact algorithm, treewidth algorithm

ACM Reference format:
Fedor V. Fomin, Daniel Lokshtanov, Sudeshna Kolay, Fahad Panolan, and Saket Saurabh. 2020. Subexponential Algorithms for Rectilinear Steiner Tree and Arborescence Problems. *ACM Trans. Algorithms* 16, 2, Article 21 (March 2020), 37 pages.
https://doi.org/10.1145/3381420

A preliminary version of this article appeared in the proceedings of SoCG 2016. This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 819416) and from the Norwegian Research Council via grant MULTIVAL. S. Saurabh was supported by Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

Authors' addresses: F. V. Fomin, Department of Informatics, University of Bergen, PB 7803, Bergen, 5020, Norway; email: fedor.fomin@uib.no; D. Lokshtanov, University of California Santa Barbara, Santa Barbara, CA; email: daniello@ucsb.edu; S. Kolay, Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, 721302, India; email: sudeshna.kolay@gmail.com; F. Panolan, Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad, Kandi, Sangareddy, 502285, India; email: fahad@iith.ac.in; S. Saurabh, Institute of Mathematical Sciences, Homi Bhabha National Institute, Theoretical Computer Science Department, IRL 2000 ReLaX, Chennai, 600113, India, and Department of Informatics, University of Bergen, PB 7803, Bergen, 5020, Norway; email: saket@imsc.res.in.

1549-6325/2020/03-ART21 $15.00
https://doi.org/10.1145/3381420

1 INTRODUCTION

In the Steiner Tree problem, we are given as input a connected graph G, a non-negative weight function $w : E(G) \to \{1, 2, \ldots, W\}$, and a set of terminal vertices $K \subseteq V(G)$. The task is to find a minimum-weight connected subgraph of G, which is a tree, containing all terminal nodes K. Steiner Tree is one of the central and best-studied problems in computer science. We refer to the books of Hwang et al. [13] and Prömel and Steger [20] for thorough introductions to the problem.

In this article, we give a deterministic *subexponential* algorithm for an important geometric variant of Steiner Tree, namely Rectilinear Steiner Tree. Here, for a given set of terminal points K in the Euclidean plane with ℓ_1-norm, the goal is to construct a network of minimum length connecting all points in K. This variant of the problem is extremely well studied. Please see Chapter 3 of the book by Brazil and Zachariasen [2] for an extensive overview of various applications of Rectilinear Steiner Tree.

For the purpose of this article, it is convenient to define Rectilinear Steiner Tree as the Steiner Tree problem on a special class of graphs called *Hanan grids*. Recall that for two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ in the Euclidean plane $\mathbb{R}^2$, the *rectilinear* (ℓ_1, *Manhattan* or *taxicab*) distance between p_1 and p_2 is $d_1(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$.

Definition 1.1 (Hanan grid [11]). Given a set $K = \{z_1, z_2, \ldots, z_n\}$ of n terminal points in the Euclidean plane $\mathbb{R}^2$, the *Hanan grid* G of K is defined as follows. The vertex set $V(G)$ of G is the set of intersection points obtained by drawing a horizontal line (line parallel to the x-axis) and a vertical line (line parallel to the y-axis) through each point of K. For every $u, v \in V(G)$, there is an edge between u and v in G, if and only if u and v are adjacent along a horizontal or vertical line; the weight of edge uv is the rectilinear distance between u and v. For a Hanan grid G, we define a weight function $\mathrm{recdist}_G$ from the edge set $E(G)$ to $\mathbb{R}$ such that for an edge $uv \in E(G)$, $\mathrm{recdist}_G(uv) = d_1(u, v)$. If the graph G is clear from the context, we drop the subscript from $\mathrm{recdist}_G$ and only use recdist.

Let us note that when G is the Hanan grid of a set K of n points, then $K \subseteq V(G)$, $|V(G)| \leq n^2$, and for every $u, v \in V(G)$, the weight of a shortest path between u and v is equal to $d_1(u, v)$. For edge $uv \in E(G)$, we say that uv is a *horizontal (vertical) edge* if both points u and v are on the same horizontal (vertical) line. It was shown by Hanan [11] that the Rectilinear Steiner Tree problem can be defined as the following variant of Steiner Tree.

Rectilinear Steiner Tree
Input: A set $K = \{z_1, z_2, \ldots, z_n\}$ of n terminal points, the Hanan grid G of K, and $\mathrm{recdist}_G$.
Output: A minimum Steiner tree for K in G.

Previous work. Although the Rectilinear Steiner Tree problem is a very special case of the Steiner Tree problem, the decision version of the problem is known to be NP-complete [9]. The detailed account of various algorithmic approaches applied to this problem can be found in books of Brazil and Zachariasen [2] and Hwang et al. [13]. In particular, several exact algorithms for this problem can be found in the literature. The classic algorithm of Dreyfus and Wagner [5] from 1971 solves Steiner Tree on general graphs in time $3^n \cdot \log W \cdot |V(G)|^2$, where W is the maximum edge weight in G. For Rectilinear Steiner Tree, an adaptation of the Dreyfus-Wagner algorithm provides an algorithm of running time $O(n^2 \cdot 3^n)$. The survey of Ganley [7] summarizes the chain of improvements based on this approach concluding with the $O(n^2 \cdot 2.62^n)$-time algorithm of Ganley and Cohoon [8]. Thomborson et al. [23] and Deneen et al. [4] gave randomized algorithms with running time $2^{O(\sqrt{n} \log n)}$ for the special case of Rectilinear Steiner Tree when the terminal points T are drawn from a uniform distribution on a rectangle. In this work, we design a deterministic algorithm to answer the question of whether Rectilinear Steiner Tree can be solved in

time subexponential in the number of terminals. The running time of our algorithm is $2^{O(\sqrt{n}\log n)}$. Smith and Wormald [22] claimed to give a $2^{O(\sqrt{n}\log n)}$ algorithm for Rectilinear Steiner Tree. However, we believe that there is an error in the claimed running time recurrence and that the correct recurrence would result in a running time of $2^{O(\sqrt{n}\log^2 n)}$. Moreover, our approach to solving the problem is very different from that given inthe work of Smith and Wormald.

It is also worth mentioning relevant parameterized algorithms for Steiner Tree on general graphs. Fuchs et al. [6] provided an algorithm with running time $O((2+\varepsilon)^n|V(G)|^{f(1/\varepsilon)}\log W)$. Björklund et al. [1] and Nederlof [17] gave $2^n|V(G)|^{O(1)}\cdot W$ time algorithms for Steiner Tree. Let us remark that, since the distances between adjacent vertices in a Hanan grid can be exponential in n, the algorithms of Björklund et al. and of Nederlof do not outperform the Dreyfus-Wagner algorithm for the Rectilinear Steiner Tree problem. Interesting recent developments also concern Steiner Tree on planar graphs, and more generally, on graphs of bounded genus. Although the existence of algorithms running in time subexponential in the number of terminals on these graph classes is still open, Pilipczuk et al. [18, 19] showed that Steiner Tree can be solved in time subexponential in the size of the Steiner tree on graphs of bounded genus.

The techniques described in this article also yield a subexponential algorithm for the Rectilinear Steiner Arborescence problem, which is closely related to Rectilinear Steiner Tree.

Definition 1.2. Let G be a graph, $K \subseteq V(G)$ be a set of terminals, and $r \in K$ be a root vertex. A *Steiner arborescence of K in G* is a subtree T of G rooted at r with the following properties:

- T contains all vertices of K, and
- for every vertex $z \in K \setminus \{r\}$, the unique path in T connecting r and z is also the shortest r–z path in G.

Let us note that if T is a Steiner arborescence of K in G, then for every vertex $v \in V(T)$, the unique path connecting r and v in T is also a shortest r–v path in G. The Rectilinear Steiner Arborescence problem is defined as follows.

Rectilinear Steiner Arborescence
Input: A set K of n terminal points, the Hanan grid G of K, a root $r \in K$, and $\mathrm{recdist}_G$.
Output: A minimum length Steiner arborescence of K.

Rectilinear Steiner Arborescence was introduced by Nastansky et al. [16] in 1974. Interestingly, the complexity of the problem was open until 2005, when Shi and Su [21] proved that the decision version of Rectilinear Steiner Arborescence is NP-complete. No subexponential algorithm for this problem was known prior to our work.

Our method. Most of the previous exact algorithms for Rectilinear Steiner Tree exploit the theorem of Hwang [12], which describes the topology of so-called full rectilinear trees. Our approach here is entirely different. The main idea behind our algorithms is inspired by the work of Klein and Marx [14], who obtained a subexponential algorithm for Subset Traveling Salesman Problem on planar graphs. The approach of Klein and Marx was based on the following two steps: (1) find a locally optimal solution such that its union with some optimal solution is of bounded treewidth, and (2) use the first step to guide a dynamic program. Although our algorithm follows this general scheme, the implementations of both steps for our problems are entirely different from those of Klein and Marx.

We give a high-level description of the algorithm for Rectilinear Steiner Tree. The algorithm for Rectilinear Steiner Arborescence is similar. In the first step, we build in polynomial time

a (possibly non-optimal) solution. To build a non-optimal Steiner tree $\widehat{T}$ of $K = \{z_1, \ldots, z_n\}$, we implement the following greedy strategy. We build $\widehat{T}$ starting from vertex z_1 and gradually connect new terminals to the tree. When we connect terminal z_{i+1} to tree T_i spanning the first i terminals, we select a shortest monotone (containing at most one "bend") path from z_{i+1} to T_i in the Hanan grid G. If there are two such paths, we select one of them according to the structure of T_i. The constructed tree $\widehat{T}$ can be seen as a "shortest path" rectilinear Steiner tree. The property of $\widehat{T}$ that is crucial for the algorithm is that there is an optimal Steiner tree T_{opt} such that graph $T = \widehat{T} \cup T_{\text{opt}}$ is of treewidth $O(\sqrt{n})$. Note that since the Hanan grid can have $O(n^2)$ vertices, from the properties of a planar graph the treewidth of the Hanan grid can be derived to be only $O(n)$ and not $O(\sqrt{n})$. Thus, this step is non-trivial.

For the second step, we have $\widehat{T}$ at hand and know that there exists a subgraph T of G of treewidth $O(\sqrt{n})$, which contains an optimal Steiner tree T_{opt} and $\widehat{T}$. Of course, if the subgraph T were given to us, then finding T_{opt} in T could be done by a standard dynamic programming on graphs of bounded treewidth. However, we only know that such a subgraph T exists, albeit with the extra information that $\widehat{T} \cup T_{\text{opt}} \subseteq T$. It turns out that this is sufficient to mimic the dynamic programming algorithm for bounded treewidth, to solve Rectilinear Steiner Tree in time $2^{O(\sqrt{n}\log n)}$.

Let us given a brief sketch of the ideas behind the dynamic programming algorithm for Steiner Tree on a rooted tree decomposition $\mathcal{T} = (\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ of the input graph (e.g., see Theorem 7.8 in Cygan et al. [3]). For each node $t \in V(\mathbb{T})$, let V_t be the union of vertices contained in all bags corresponding to nodes of the subtree of $\mathbb{T}$ rooted at t and let S_t be the subgraph induced by V_t. Then, in the dynamic programming algorithm, for each t we store a set of states, capturing the interaction between a minimal Steiner tree and subgraph S_t, particularly the weight of a tree and the information about its connected components in S_t. It is possible to ensure that all the information carried out in each state is "locally" defined—for instance, the information can be encoded by the elements of the bag X_t only. Therefore, at the root node, there is a state that corresponds to an optimal Steiner tree.

In our algorithm, we define *types*, which are analogous to the states stored at a node of a tree decomposition. A type stores all of the information of its corresponding state. Since we do not know the tree decomposition $\mathcal{T}$, a type stores more "local" information, to take care of the lack of definite information about $T = T_{\text{opt}} \cup \widehat{T}$. We guess some structural information about the virtual tree decomposition $\mathcal{T}$ of T. For example, we guess the height h of the rooted tree $\mathbb{T}$. In a rooted tree decomposition, the level of a node t is defined by the height of the subtree rooted at t. In our algorithm, we generate types over h levels. The intuition is that, for a node $t \in \mathbb{T}$ of level h', for each state of t, that was required for the dynamic programming over $\mathcal{T}$, there is an equivalent type generated in level h' of our algorithm. This implies that, at level h, there is a type equivalent to a state that corresponds to an optimal Steiner tree in T. In fact, we show that any Steiner tree corresponds to exactly one type $\widehat{D}$. During the iterative generation of types, the type $\widehat{D}$ may be generated many times. One such generation corresponds to an optimal solution. Thus, the final step of the algorithm involves investigating all occurrences of type $\widehat{D}$ in the iterative generation and finding the weight of a minimum Steiner tree. As in dynamic programming, a backtracking step will enable us to retrieve a minimum Steiner tree of $T = T_{\text{opt}} \cup \widehat{T}$ and therefore of G.

2 PRELIMINARIES

For a positive integer n, we use the notation $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For a point u in the Euclidean plane $\mathbb{R}^2$, its position is denoted by the coordinates (u_x, u_y). For a set V, a partition $\mathcal{P}$ of V is a family of subsets of V such that the subsets are pairwise disjoint and the union of the subsets is V. Each subset of a partition is called a *block*. Given two partitions $\mathcal{P}_1$ and $\mathcal{P}_2$, the join

operation results in the partition $\mathcal{P}$, which is the most refined partition of V such that each block of $\mathcal{P}_1$ and $\mathcal{P}_2$ is contained in a single block of $\mathcal{P}$. The resultant partition $\mathcal{P}$ is often denoted as $\mathcal{P}_1 \sqcup \mathcal{P}_2$. Given a block B of $\mathcal{P}$, $\mathcal{P} \setminus B$ denotes removing the block B from $\mathcal{P}$.

Given a graph G, its vertex and edge sets are denoted by $V(G)$ and $E(G)$, respectively. For a vertex $v \in V(G)$, the degree of v, denoted as $\text{degree}_G(v)$, is the number of edges of $E(G)$ incident with v. Given a vertex subset $V' \subseteq V(G)$, the induced subgraph of V', denoted by $G[V']$, is the subgraph of G, with V' as the vertex set and the edge set defined by the set of edges that have both endpoints in V'. An edge between two vertices u and v is denoted as uv. A path where vertices $\{u_1, u_2, \ldots, u_\ell\}$ appear in sequence is denoted by $u_1u_2 \cdots u_\ell$. Similarly, a path where vertices u and v are the endpoints is called a *u–v path*. Given a path P its non-endpoint vertices are referred to as internal vertices. For an edge uv, an edge contraction in G results in a graph G' defined as follows. The vertex set $V(G') = (V(G) \setminus \{u, v\}) \cup v_{new}$, where v_{new} is a new vertex. The edge set $E(G') = E(G[V(G) \setminus \{u, v\}]) \cup \{wv_{new} | wu \in E(G) \vee wv \in E(G)\}$. By $G' \leq_s G$, we mean that the graph G' is a subgraph of G. Given a graph G and a vertex (edge) set S, we denote by $G - S$ the subgraph resulting from G after deleting the vertices (edges) in S. Given a weight function $w : E(G) \to \mathbb{R}$, for a subgraph H of G, we use $w(H)$ to denote the number $\sum_{e \in E(H)} w(e)$. Furthermore, for two vertices s and t in $V(G)$, by the term *shortest path* between s and t we mean a shortest path with respect to the weight function w. Given two subgraphs G_1, G_2 of G, a shortest path between G_1 and G_2 is a path P between a vertex $u \in V(G_1)$ and a vertex $v \in V(G_2)$ such that, among the shortest paths for each possible pair $\{u' \in V(G_1), v' \in V(G_2)\}$, P has minimum length. Given two graphs G_1 and G_2, the union graph $G_1 \cup G_2$ has vertex set $V(G_1 \cup G_2) = V(G_1) \cup V(G_2)$, whereas the edge set $E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$.

2.1 Treewidth

Let G be a graph. A *tree-decomposition* of a graph G is a pair $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$, where $\mathbb{T}$ is a tree whose every node $t \in V(\mathbb{T})$ is assigned a subset $X_t \subseteq V(G)$, called a *bag*, such that the following conditions hold:

- $\bigcup_{t \in V(\mathbb{T})} X_t = V(G)$;
- for every edge $xy \in E(G)$, there is a $t \in V(\mathbb{T})$ such that $\{x, y\} \subseteq X_t$; and
- for any $v \in V(G)$, the subgraph of $\mathbb{T}$ induced by the set $\{t \mid v \in X_t\}$ is connected.

The *width* of a tree decomposition is $\max_{t \in V(\mathbb{T})} |X_t| - 1$. The *treewidth* of G is the minimum width over all tree decompositions of G and is denoted by $\text{tw}(G)$. A tree decomposition $\mathcal{T} = (\mathbb{T}, \mathcal{X})$ is called a *nice tree decomposition* if $\mathbb{T}$ is a tree rooted at some node r where $X_r = \emptyset$, each node of $\mathbb{T}$ has at most two children, and each node is of one of the following kinds:

- *Introduce node:* A node t that has only one child t' where $X_t \supset X_{t'}$ and $|X_t| = |X_{t'}| + 1$. Note that the new vertex is introduced as an isolated vertex, and no edges incident to it are introduced in this bag.
- *Introduce edge node:* A node t labeled with an edge uv, with only one child t' such that $\{u, v\} \subseteq X_{t'} = X_t$. This bag is said to introduce uv.
- *Forget vertex node:* A node t that has only one child t' where $X_t \subset X_{t'}$ and $|X_t| = |X_{t'}| - 1$.
- *Join node:* A node t with two children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$.
- *Leaf node:* A node t that is a leaf of $\mathbb{T}$, and $X_t = \emptyset$.

We additionally require that every edge is introduced exactly once.

Given the tree decomposition $\mathcal{T}$, consider the the underlying tree $\mathbb{T}$. For any node $t \in V(\mathbb{T})$, we define the *level* of t as the height of the subtree rooted at t. Here, the *height* of a node t is the number of vertices in the longest downward path to a leaf from that node.

Proposition 2.1 (Kloks [15]). *Given a graph G and a tree decomposition $\mathcal{T}'$ rooted at a node r' and of treewidth w, there is a polynomial time algorithm that finds a nice tree decomposition $\mathcal{T} = (\mathbb{T}, \mathcal{X})$, rooted at a node r and of treewidth w, such that the height of the underlying tree $\mathbb{T}$ is at most $3(|V(G)| + |E(G)|)$.*

2.2 Planar Graph Embeddings and Minors

A graph is *planar* if it can be embedded in the plane. In other words, it can be drawn on the plane in such a way that its edges intersect only at their endpoints. Formally, a planar embedding Π of a graph G consists of an injective mapping $\Pi : V(G) \to \mathbb{R}^2$ and a mapping Π of edges $uv \in E(G)$ to simple curves in $\mathbb{R}^2$ that join $\Pi(u)$ and $\Pi(v)$. In addition, for $e, f \in E(G)$, $\Pi(e) \cap \Pi(f)$ contains only the images of common end vertices, and for $e \in E(G)$ and $v \in V(G)$, $\Pi(v)$ is not an internal point of $\Pi(e)$. Now we define the notion of a minor of a graph G.

Definition 2.2. A graph H is a minor of a graph G, denoted as $H \leq_m G$, if it can be obtained from a subgraph of G by a sequence of edge contractions.

Notice that this implies that H can be obtained from G by a sequence of vertex deletions, followed by a sequence of edge deletions and finally a sequence of edge contractions. We will need the following folklore observation.

Observation 1. *Suppose G, H are connected graphs such that H is a minor of G. Then H can be obtained from G only by a sequence of edge deletions and contractions.*

We also will be using the notion of a minor model.

Definition 2.3. Let G and H be two connected graphs, and $H \leq_m G$. A *minor model* or simply a *model* of the graph H is a collection of pairwise disjoint vertex subsets $\mathcal{P}(H) = \{C_v \subseteq V(G) \mid v \in V(H)\}$ such that

(a) $V(G) = \biguplus_{v \in V(H)} C_v$;
(b) for each $v \in V(H)$, $G[C_v]$ is connected; and
(c) for any $uv \in E(H)$, there exists $w \in C_u$ and $w' \in C_v$ such that $ww' \in E(G)$.

Remark 1. It is important to point out that in general the definition of minor model does not demand that the vertex sets in $\mathcal{P}(H) = \{C_v \subseteq V(G) \mid v \in V(H)\}$ form a partition of $V(G)$. However, when both G and H are connected, one can easily show that even this extra property can be assumed.

Given a planar graph G with an embedding Π, we call the vertices of the outer face *boundary vertices* of the embedding Π and all other vertices *internal vertices* of the embedding Π.

Definition 2.4. Let G be a planar graph with a planar embedding Π, and let C be a simple cycle of G. Let p_∞ be a point in the outer face of G in the embedding Π. Then removal of C from $\mathbb{R}^2$ divides the plane into two regions. The region that does not contain the point p_∞ is called the *internal region* of C, and the region containing p_∞ is called the *outer/external region* of C. A vertex in $V(G)$ is called *internal with respect to* C if it lies in the internal region of C, and *external with respect to* C if it lies in the external region of C. An edge in $E(G) \setminus E(C)$ is called an *external edge* if all points on its curve lie in the external region of C. Otherwise, an edge in $E(G) \setminus E(C)$ is called an *internal edge*.

Therefore, an edge of $E(G)$ can be exactly one of the three kinds: an edge of C, an external edge, or an internal edge. Similarly, a vertex can be exactly one of the three kinds: a vertex of C, an external vertex with respect to C, or an internal vertex with respect to C.

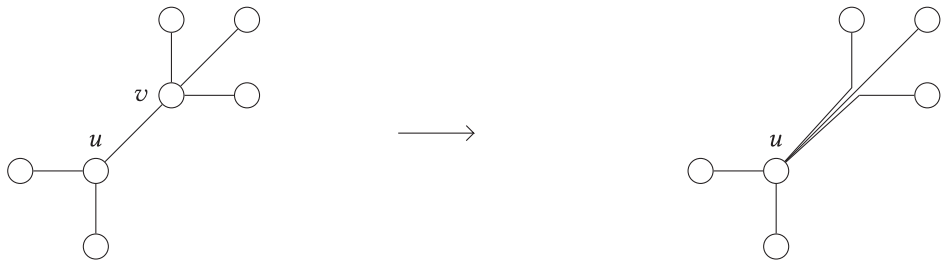

Fig. 1. Derived embedding: the edge uv is contracted to the vertex u.

Observation 2. *Let G be a planar graph with a planar embedding Π in $\mathbb{R}^2$. Let p_∞ be a point in the outer face of Π. Let H be a minor of G, and let $\mathcal{P}(H) = \{C_v | v \in V(H)\}$ be a minor model of H. Then H is a planar graph. Furthermore, a planar embedding Π' of H can be obtained from Π that satisfies the following properties:*

- *Every vertex $v \in H$ is positioned in the place of a vertex in C_v.*
- *The point p_∞ is on the outer face of Π'.*

We call such a planar embedding Π' the embedding derived *from* Π.

Proof. It is well known that a minor of a planar graph is also a planar graph. We modify Π to obtain Π' in the following way. Let $O = \{o_1, o_2, \ldots, o_\ell\}$ be the sequence of edge deletions and edge contractions. Starting from $G_0 = G$ and ending at $G_\ell = H$, each operation o_i creates a new graph G_i. We will induct over the length of the sequence O to show that at the end of each operation, we obtain a planar embedding Π_i of G_i derived from the planar embedding Π of G. In the base case, we obtain a planar embedding Π_1 for G_1 derived from $\Pi_0 = \Pi$ as follows:

(1) If o_1 is a deletion operation, then it changes the embedding of the current graph by simply deleting the element concerned. No other vertex or edge changes position.
(2) Suppose o_1 contracts the edge uv. Let the degree of v be d. We first make $d - 1$ parallel copies of uv. We forget the vertex v and the original edge uv. We reroute all other edges incident with v using the original edges in Π and one of the $d - 1$ copies of uv. See Figure 1. The new embedding is a planar embedding of the new graph.

Thus, the embedding Π_1 ensures the two preceding properties and is a derived embedding from Π.

In the induction step $i \leq \ell$, we have a planar embedding Π_{i-1} for graph G_{i-1} where (i) every vertex $v \in G_{i-1}$ is positioned in the place of a vertex in C_v and (ii) the point p_∞ is on the outer face of Π_{i-1}. In other words, the embedding Π_{i-1} for G_{i-1} is derived from the embedding Π for G. To obtain an embedding Π_i for G_i, we do as follows:

(1) If o_i is a deletion operation, then it changes the embedding Π_{i-1} by simply deleting the element concerned. No other vertex or edge changes position.
(2) Suppose o_i contracts the edge uv. Let the degree of v be d. We first make $d - 1$ parallel copies of uv. We forget the vertex v and the original edge uv. We reroute all other edges incident with v using the original edges in embedding Π_{i-1} and one of the $d - 1$ copies of uv (see Figure 1). The new embedding Π_i is a planar embedding of the G_i.

This ensures that Π_i is derived from Π_{i-1}. Since Π_{i-1} was derived from Π, by definition this ensures that Π_i is a derived embedding from Π. Therefore, $\Pi_\ell = \Pi'$ for H is a derived embedding from Π for G. □

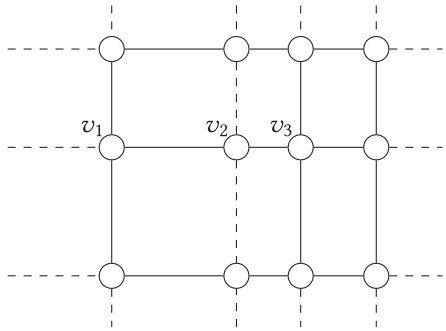

Fig. 2. The solid edges define a subgrid of a grid. The vertex v_1 is a boundary vertex of the subgrid. The vertex v_2 is a subdivision vertex, and v_3 is a cross vertex; both v_2 and v_3 are also interior vertices of the subgrid.

Grids and subgrids play an important role in this article. For $n \in \mathbb{N}$ and a subset $W \subseteq [n]$, by $\max W$ ($\min W$) we denote the maximum (minimum) element of W.

Definition 2.5. Let n, m be two positive integers. An $n \times m$ grid is a graph G such that $V(G) = \{v_{i,j} \mid i \in [n], j \in [m]\}$ and $E(G) = \{v_{ij}v_{i'j'} \mid |i - i'| + |j - j'| = 1\}$. For any $i \in [n]$, we call $\{v_{i1}, \ldots, v_{im}\}$ to be the i-th row of the grid G, and for any $j \in [m]$, we call $\{v_{1j}, \ldots, v_{nj}\}$ to be the j-th column of the grid G. The vertices in the first row and n-th row and the first column and m-th column are called the *boundary vertices* of the grid. The vertices that are not boundary vertices are called *interior vertices* of the grid.

The graph H is called a *subgrid* of G if there exist subsets $R \subseteq [n], C \subseteq [m]$ such that $V(H) = \{v_{ij} \in V(G) : (\min R \leq i \leq \max R) \wedge (\min C \leq j \leq \max C) \wedge (i \in R \vee j \in C)\}$ and $E(H) = \{v_{ij}v_{i'j'} \in E(G) : v_{ij}, v_{i'j'} \in V(H) \wedge (i = i' \in R \vee j = j' \in C)\}$. The set of vertices $\{v_{ij} \in V(H) : i \notin \{\min R, \max R\} \vee j \notin \{\min C, \max C\}\}$ are called the *interior vertices* of H. The set of vertices $\{v_{ij} \in V(H) : i \in R \wedge j \in C\}$ are called *cross vertices*. Finally, the set of vertices $\{v_{ij} \in V(H) : i \notin R \vee j \notin C\}$ are called *subdivision vertices* of H. See Figure 2.

For the proof of the first step of the algorithm, we will use the following auxiliary lemma.

Lemma 2.6. *Let G and H be two connected planar graphs such that $H \leq_m G$, and let $\mathcal{P}(H) = \{C_v | v \in V(H)\}$ be a minor model of H in G. Let also Π' be an embedding of H derived from a planar embedding Π of G. Suppose H contains an induced subgraph H' isomorphic to a 3×3 grid. Let C' be the cycle formed by boundary vertices of H', and let v be the vertex of H' in the internal region of C'. Then there is a simple cycle C in G such that*

(1) $V(C) \subseteq \bigcup_{u \in V(H'); u \neq v} C_u$.
(2) For each vertex $w \in G$ that is contained in the internal region of C in Π, there is a vertex $u \in V(H')$ with $w \in C_u$.
(3) All vertices of C_v are completely contained in the internal region of C in Π.
(4) There is a vertex $w \in C_v$ such that $\mathsf{degree}_G(w) \geq 3$.

Proof. Consider consecutive vertices u, w in C'. The edge uw corresponds to an edge $u'w' \in E(G)$ such that $u' \in C_u, w' \in C_w$. We will call edges like $u'w'$ *marked edges* in G. Note that both u' and v' do not belong to C_v. Now, for a boundary vertex u of H', consider the connected subgraph $G[C_u]$ of G. There are at most two vertices, u_1 and u_2, that are incident with marked edges in C_u. Since $G[C_u]$ is a connected graph, let $P_{u_1u_2}$ be a path connecting u_1 and u_2 in $G[C_u]$. We call $P_{u_1v_2}$ a *marked path*. Note that $V(P_{u_1u_2}) \cap C_v = \emptyset$. The union of the marked edges and marked paths

forms the simple cycle C, and the vertex set $V(C)$ is disjoint from C_v. This proves condition (1). In fact, a vertex of C only belongs to C_u for a vertex $u \in V(C')$.

Now, we show condition (2). Consider a vertex w that is contained in the internal region of C in Π. Since G and H are both connected graphs, by Definition 2.3, there is a vertex $u \in V(H)$ such that $w \in C_u$. If $u \in V(H')$, then condition (2) holds. Suppose not. Then u belongs to the external region of C' in Π'. By Observation 2, u is positioned at a vertex $w' \in C_u$. This means that C_u has a vertex w' in the external region of C and a vertex w in the internal region of C. Since $u \notin V(H')$, $C_u \cap V(C) = \emptyset$. However, $G[C_u]$ is a connected subgraph of G and cannot be embedded without crossing with the cycle C. This is a contradiction to the fact that G is a planar graph. Hence, condition (2) must hold.

Next, we show that for the interior vertex $v \in V(H')$, all vertices of C_v are completely contained in the internal region of C. From the definition of the derived embedding Π' from Π, as described in Observation 2, the point p_∞ is a point in the outer face of both embeddings. The interior vertex $v \in V(H')$ is contained in the internal region of C'. Then, from the definition of Π' derived from Π, v is placed in the position of a vertex $u \in C_v$. By construction of C, it is disjoint from the vertices of C_v. Since $G[C_v]$ is connected, to maintain planarity, it follows that C_v is in the internal region of C. This shows that condition (3) must hold.

Last, we show that there is a vertex $w \in C_v$ that has at least three neighbors in G. Notice that the induced subgraph G' of G, formed by the vertices of C and the internal region of C, also has H' as a minor. For a vertex $u \in V(H')$, let D_u denote the restriction of C_u in G'. Since, for the interior vertex $v \in V(H')$, C_v is completely contained in the internal region of C, $D_v = C_v$. There are four neighbors of v in H'. For a neighbor u of v, let e be an edge between D_u and C_v. We call the endpoint of e, in C_v, a *marked vertex*. There are at most four marked vertices in C_v.

Suppose there are at most two marked vertices. This means that at least one marked vertex, say w, has at least two neighbors outside C_v. Since C_v is connected, and there are at least two marked vertices in C_v, the degree of w must be at least 3 in G.

Otherwise, there are at least three marked vertices $x_1, x_2, x_3 \in C_v$. Let P_{12} be a shortest path in C_v, between x_1 and x_2. Among the vertices of P_{12}, let w be the closest to x_3, and let Q be the shortest path between w and x_3. First, if $w = x_3$, then w must be internal in P_{12} and thus has at least two neighbors in C_v. In this case, since w is also a marked vertex, it has a third neighbor outside C_v, thereby making its degree in G at least 3. Otherwise, when $w \neq x_3$, suppose w is an internal vertex of P_{12}. Then, the two neighbors in P_{12} and a neighbor in Q makes the degree of w at least 3 in G. Otherwise, w is one of x_1 or x_2. This means that w has a neighbor in P_{12}, x_3 as a neighbor, and a neighbor outside C_v. Thus, w has degree at least 3 in G. This completes the proof. □

Finally, our proof will also need the following planar grid-minor theorem.

Proposition 2.7 (Gu and Tamaki [10]). *Lett be a non-negative integer. Then every planar graph G of treewidth at least $9t/2$ contains a $t \times t$ grid as a minor.*

2.3 Properties of Shortest Paths in the Hanan Grid

Let G be the Hanan grid of a set of n points P. For a subgraph H of G and $v \in V(H)$, we say that v is a *bend vertex* if there exists at least one horizontal edge and at least one vertical edge from $E(H)$ incident with the vertex v. A path $R = u_1 \cdots u_\ell$, between u_1 and u_ℓ in G, is called a *monotone path* if there exists $i \in [\ell]$ such that the points $u_1, \ldots, u_i$ belong to a horizontal line and $u_i, \ldots, u_\ell$ belong to a vertical line or vice versa. In other words, all horizontal edges and all vertical edges in R are contiguous.

The following observation contains some simple facts about monotone paths.

Observation 3. *Let u and v be two vertices of a Hanan grid G. Then,*

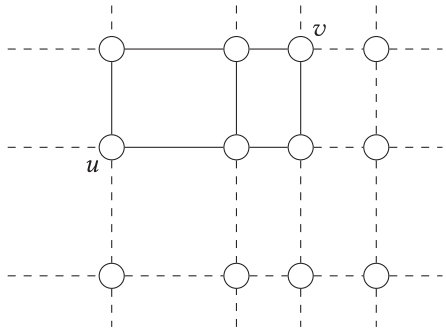

Fig. 3. The solid lines represent the grid defined by vertices u and v as its diagonal points.

(a) *There is at least one and at most two monotone u–v paths.*
(b) *If the x-coordinates of u and v are equal, then there is only one monotone u–v path and all edges in this path are vertical. Similarly, if the y-coordinates of u and v matches, the unique monotone u–v path consists of horizontal edges only.*
(c) *If there are two monotone paths between u and v, then one path has a horizontal edge incident with u and the other path has a vertical edge incident with u.*

Definition 2.8. Suppose we are given a Hanan grid G of a set of terminals K and two vertices $u, v \in V(G)$. Let $x_1 = \min\{u_x, v_x\}$, $x_2 = \max\{u_x, v_x\}$, $y_1 = \min\{u_y, v_y\}$, and $y_2 = \max\{u_y, v_y\}$. Let $V' = \{w \in V(G) | w_x \in [x_1, x_2], w_y \in [y_1, y_2]\}$. Then $G' = G[V']$, the subgraph of G induced by V', is called a *grid* defined by the two vertices u, v as its diagonal points. See Figure 3.

Observation 4. *Given a Hanan grid G, a shortest path between any two vertices u, v has the property that the sequence of the x-coordinates of the vertices of the path is a monotone sequence and the sequence of their y-coordinates is also a monotone sequence.*

Observation 5. *Given a grid G, all shortest paths between two vertices u, v are contained in the grid $G' \leq_s G$ that is defined by u, v as its diagonal points. In fact, any path, with the property that the sequence of the x-coordinates of the vertices of the path is a monotone sequence and the sequence of their y-coordinates is also a monotone sequence, and which is fully contained inside G', is a shortest path between u and v.*

3 SUBEXPONENTIAL ALGORITHM FOR RECTILINEAR STEINER TREE

In this section, we give a subexponential algorithm for Rectilinear Steiner Tree. Let K be an input set of terminals (points in $\mathbb{R}^2$), $|K| = n$, and G be the Hanan grid of K. Furthermore, let recdist_G denote the weight function on the edge set $E(G)$. For brevity, we will use recdist for recdist_G. The described algorithm is based on a dynamic programming over vertex subsets of size $O(\sqrt{n})$ of G. To reach the stage where we can apply the dynamic programming algorithm, we do as follows. First, we define a rectilinear Steiner tree, called *shortest path RST*, and describe some of its properties. Next, we show that for a shortest path RST $\widehat{T}$, there is an optimal Steiner tree T_{opt} such that $\widehat{T} \cup T_{\text{opt}}$ has treewidth of $O(\sqrt{n})$. Finally, keeping a hypothetical tree decomposition of $\widehat{T} \cup T_{\text{opt}}$ in mind, we design a dynamic programming algorithm to obtain the size of a minimum rectilinear Steiner tree of G.

3.1 Shortest Path RST and Its Properties

In this part, we define a shortest path RST for a set $K = \{z_1, \ldots, z_n\}$ of input terminals and prove some useful properties of such a Steiner tree. We define a *shortest path RST* as follows.

Let G be the Hanan grid of K. We define a shortest path RST $\widehat{T}$ through the following constructive greedy process. Initially, we set T_1 to the graph $(\{z_1\}, \emptyset)$, which is a rectilinear Steiner tree of $\{z_1\}$. In the i-th step, we compute a rectilinear Steiner tree T_{i+1} of $\{z_1, \dots z_{i+1}\}$ from T_i as follows. If $z_{i+1} \in V(T_i)$, then we set $T_{i+1} = T_i$. Otherwise, let v be a vertex in T_i such that $\mathrm{recdist}(v, z_{i+1}) = \min_{u \in V(T_i)} \mathrm{recdist}(u, z_{i+1})$. If there is only one monotone $z_{i+1} - v$ path, then let Q be this path. Otherwise, there are two monotone $z_{i+1} - v$ paths, such that one path has a horizontal edge incident with v and the other has a vertical edge incident with v. If there is a horizontal edge in T_i that is incident with v, then we choose Q to be the monotone $z_{i+1} - v$ path such that the edge in Q incident with v is a horizontal edge. Otherwise, we choose Q to be the monotone $z_{i+1} - v$ path such that the edge in Q incident with v is a vertical edge. Then we construct T_{i+1} by adding the chosen monotone path Q to T_i. After $n - 1$ iterations, we construct a tree $\widehat{T} = T_n$ of G, which is a Steiner tree of K. This is our shortest path RST.

It is easy to see that one can construct a shortest path RST in polynomial time.

Lemma 3.1. *Given a set K of terminal points and the Hanan grid G of K, a shortest path RST $\widehat{T}$ of K can be constructed in polynomial time.*

Proof. Consider the procedure used to define a shortest path RST. The procedure involves $|K|$ steps. In each step, polynomially many shortest paths from one terminal to a set of vertices of G are found out. Since this step can be executed in polynomial time, and there are polynomially many steps, the construction of $\widehat{T}$ requires polynomial time. □

Next, we give an upper bound on the number of bend vertices in a shortest path RST. Recall that for a subgraph H of the Hanan grid G, a vertex $v \in V(H)$ is a bend vertex if there is at least one horizontal edge and at least one vertical edge in $E(H)$ incident to v.

Lemma 3.2. *The number of bend vertices in $\widehat{T}$ is at most n.*

Proof. We prove the assertion by induction on the number of iterations to construct the solution $\widehat{T}$. Toward this, using induction on i, we prove that the number of bend vertices in T_i is at most i. In the base case, T_1 is a single vertex with no bend vertices. Suppose the statement holds for T_{i-1}. If z_i is already contained in T_{i-1}, then $T_i = T_{i-1}$. By induction, the number of bend vertices in T_{i-1} is at most $i - 1$, and therefore the number of bend vertices in T_i is at most i. Otherwise, we find a vertex $v \in V(\widehat{T}_{i-1})$ such that a shortest path Q between z_i and $\widehat{T}_{i-1}$ ends with v.

Since Q is a shortest path between z_i and $\widehat{T}_{i-1}$, the set of internal vertices of Q is disjoint from $V(\widehat{T}_{i-1})$. By induction hypothesis, T_{i-1} had at most $i - 1$ bend vertices. The number of bend vertices in Q is at most 1. If v is already a bend vertex in T_{i-1}, then the number of bend vertices in T_i is at most $i - 1 + 1 = i$. By the way, we define shortest path RST such that if v is not a bend vertex in T_{i-1}, it is also not a bend vertex in T_i. Therefore, in this case, the number of bend vertices in T_i, if Q has a bend vertex, is also at most i. This concludes the proof. □

3.2 Supergraph of an Optimal RST with Bounded Treewidth

We view the Hanan grid G as a planar graph and use this viewpoint to obtain the required upper bound on the treewidth of a subgraph of G. In particular, given a shortest path RST $\widehat{T}$, we show the existence of an optimal Steiner tree T_{opt} such that the treewidth of $\widehat{T} \cup T_{\mathrm{opt}}$ is $O(\sqrt{n})$. First, we show that there is an optimal Steiner tree in G that has a bounded number of bends.

Lemma 3.3. *Let K be a set of n points in $\mathbb{R}^2$ and G be the Hanan grid of K. Then there is an optimal rectilinear Steiner tree of K such that the number of bend vertices in the rectilinear Steiner tree is at most $3n$.*

Proof. We prove the lemma using induction on $n = |K|$. The base cases are when $n = 1, 2$. Since the tree $(z_1, \emptyset)$ is an optimal Steiner tree when $|K| = 1$, the number of bend vertices in the tree $(z_1, \emptyset)$ is zero. Similarly, when $n = 2$, a monotone path between the two terminal vertices is an optimal Steiner tree. Therefore, the number of bends is at most 1 and the hypothesis is still true.

Consider the induction step where $n = |K| > 2$. Let T_{opt} be an optimal Steiner tree of K in G. Since T_{opt} is an optimal Steiner tree of K, there are at least two leaves, and each leaf node is a terminal. In addition, there is a pair $\{z_1, z_2\}$ of leaf terminals such that, in the $z_1 - z_2$ path P of T_{opt}, there is at most one internal vertex with degree at least 3 in T_{opt}. This means that all other internal vertices in P are of degree exactly 2 in T_{opt}. If there are no internal vertices of degree at least 3 in P, this means that all terminals in K are collinear and that T_{opt} is a path. Otherwise, let u be an internal vertex of P with degree at least 3 in T_{opt}. Consider the subpaths P_1 and P_2, which are $z_1 - u$ and $z_2 - u$ paths. Suppose there is a terminal z appearing as an internal vertex on P. Without loss of generality, we can assume that $z \in V(P_1)$ and z is the second terminal vertex in the path P_1 (the first terminal vertex is z_1). Let us denote the $z_1 - z$ subpath of P_1 as P_3. By definition, all internal vertices of P_3 are degree 2 non-terminal vertices. Let T_1 be the tree obtained by deleting $V(P_3) \setminus \{z\}$ from T_{opt}. Since T_{opt} is an optimal Steiner tree of K, T_1 is an optimal Steiner tree of $T \setminus \{z_1\}$ and P_3 is a minimum weight $z_1 - z$ path. Since $|K \setminus \{z_1\}| = n - 1$, by induction hypothesis, there is an optimal Steiner tree T' of $K \setminus \{z_1\}$ such that the number of bend vertices is at most $3(n-1)$. Let P_3' be a monotone $z_1 - z$ path. Since $\text{recdist}(T' \cup P_3') \leq \text{recdist}(T_1 \cup P_3) = \text{recdist}(T_{\text{opt}})$, $T' \cup P_3'$ is an optimal Steiner tree of K. By the definition of a monotone path, the number of bend vertices in P_3' is at most 1. Thus, any bend vertex b in $T' \cup P_3'$ is a bend vertex in T', or a bend vertex in P_3', or $b = z$. This implies that the number of bend vertices in $T' \cup P_3'$ is at most $3(n-1) + 1 + 1 \leq 3n$.

The remaining case is that the $z_1 - z_2$ path P has exactly one internal vertex u of degree at least 3, whereas all other internal vertices are of degree 2 and are not terminals. Let T_1 be the tree obtained by deleting $V(P_1 \cup P_2) \setminus \{u\}$ from T_{opt}. Since T_{opt} is an optimal Steiner tree of K, T_1 is an optimal Steiner tree of $K \setminus \{z_1, z_2\} \cup \{u\}$. Again, the subpaths P_1 and P_2 must be minimum weight $z_1 - u$ and $z_2 - u$ paths, respectively. Since $|K \setminus \{z_1, z_2\} \cup \{u\}| = n - 1$, by induction hypothesis, there is an optimal Steiner tree T' of $K \setminus \{z_1, z_2\} \cup \{u\}$ such that the number of bend vertices is at most $3(n-1)$. By optimality of T_{opt}, for any minimum weight $z_1 - u$ path P_1' and $z_2 - u$ path P_2', $\text{recdist}(T' \cup P_1' \cup P_2') = \text{recdist}(T_{\text{opt}})$. Thus, $T = T' \cup P_1' \cup P_2'$ is an optimal Steiner tree of K. We choose P_1' and P_2' to be monotone paths. Thus, the number of bend vertices in P_1' and P_2' is at most 1 each. This means that u could be a new bend vertex in T, and there could be at most two new bend vertices in the two monotone paths added. This brings the total number of newly introduced bend vertices to at most 3. Thus, the total number of bend vertices in T is at most $3n$. This completes the proof. □

Next, with respect to a shortest path RST $\widehat{T}$ of G, we prove that there is an optimal Steiner tree T_{opt} that, because of its carefully chosen conditions, ensures the treewidth of $T = \widehat{T} \cup T_{\text{opt}}$ is at most $41\sqrt{n}$. To get the desired upper bound on the treewidth of T, we show that it does not contain $O(\sqrt{n}) \times O(\sqrt{n})$ grid as a minor. In fact, we prove that if there is a large grid, then we can find a "clean part of the grid" (subgrid not containing vertices of K and bend vertices of either $\widehat{T}$ or T_{opt}) and reroute some of the paths in either $\widehat{T}$ or T_{opt}. This, in turn, contradicts either the way $\widehat{T}$ is constructed or the optimality of T_{opt}.

Lemma 3.4. *Given a set K of n points and a shortest path RST $\widehat{T}$ of K, there is an optimal rectilinear Steiner tree T_{opt} of K with the property that the treewidth of $\widehat{T} \cup T_{\text{opt}}$ is at most $41\sqrt{n}$.*

Proof. Among the optimal Steiner trees of K with the minimum number of bend vertices, we select a tree T_{opt} that has maximum edge intersection with $E(\widehat{T})$. From Lemma 3.3, it follows that the number of bend vertices in T_{opt} is at most $3n$.

Let $T = \widehat{T} \cup T_{\text{opt}}$. Let $\widehat{B}$ and B_{opt} be the set of bend vertices in $\widehat{T}$ and T_{opt}, respectively. Let $U = K \cup \widehat{B} \cup B_{\text{opt}}$ and $N = V(G) \setminus U$. Since $|K| = n$, $|\widehat{B}| \leq n$, and $|B_{\text{opt}}| \leq 3n$, we know that $|U| \leq 5n$. Let Π_T be a planar embedding of T, obtained by deleting all edges and vertices not in T from the planar embedding Π of G. We show that the treewidth of T is at most $41\sqrt{n}$. We can assume that $n \geq 4$, as otherwise we can greedily find out the best rectilinear Steiner tree from the constant-sized Hanan grid. For the sake of contradiction, assume that $\text{tw}(T) > 41\sqrt{n}$. Then, by Proposition 2.7, there is a $9\sqrt{n} \times 9\sqrt{n}$ grid H appearing as a minor of T. Let $\mathcal{P}(H) = \{C_v | v \in V(H)\}$ be a minor model of H. Since H and G are connected graphs, $\mathcal{P}(H)$ is a partition of the vertex set $V(G)$. We identify a 3×3 subgrid H' of H by the following process. For any $v \in V(H)$, we mark the vertex v if $C_v \cap U \neq \emptyset$ (i.e, C_v contains a terminal or a bend vertex from $\widehat{T}$ or T_{opt}). Since $|U| \leq 5n$, the number of marked vertices in H is at most $5n$. Since H is a $9\sqrt{n} \times 9\sqrt{n}$ grid, there are at least $9n$ vertex disjoint 3×3 subgrids in H. Since there are at most $5n$ marked vertices, this implies that there is a 3×3 subgrid H' in H such that each vertex of H' is unmarked. The fact that for $u \in V(H')$, $C_u \cap U = \emptyset$ implies the following observation. □

Observation 6. *Let $u \in V(H')$ and $w \in C_u$:*

(i) $d_{\widehat{T}}(w), d_{T_{\text{opt}}}(w) \in \{0, 2\}$. If for any $T_i \in \{\widehat{T}, T_{\text{opt}}\}$, $d_{T_i}(w) = 2$, then the two edges in T_i incident with w are of the same kind (either horizontal or vertical).

(ii) If one horizontal (vertical) edge incident with w is present in $T = \widehat{T} \cup T_{\text{opt}}$, then the other horizontal (vertical) edge incident with w is also present in T. Hence, $d_T(w) \in \{2, 4\}$.

Note that H' is a connected graph and is a minor of a connected graph T. Let $\Pi_{H'}$ be a planar embedding derived from Π_T. By Lemma 2.6, we know that there is a simple cycle C' in T with the following properties:

(1) $V(C') \subseteq \bigcup_{u \in V(H')} C_u$.
(2) For each vertex $w \in G$ that is contained in the internal region of C' in Π, there is a vertex $u \in H'$ with $w \in C_u$. In particular, all vertices of $V(S) \setminus \bigcup_{u \in V(H')} C_u$ (which includes U) are not in the internal region of C'.
(3) For the interior vertex $v \in V(H')$, all vertices in C_v are in the internal region of C'.
(4) Finally, there is a vertex $w \in C_v$, where v is the interior vertex of H', in the internal region of C', such that $d_T(w) \geq 3$. By Observation 6, $d_T(w) = 4$.

In other words, there is a cycle C' in the Hanan grid G such that $V(C') \subseteq V(T) \setminus U$, every point in the internal region of C' does not correspond to any vertex in U and there is a vertex w of degree 4 in T, which is in the internal region of C'. The following claim follows from Observation 6.

Claim 1. *Let $u, v \in V(G)$ be points that are either on the same horizontal line or on the same vertical line. If the line segment L connecting u and v does not intersect with the outer region of C', and there is an edge $v_1v_2 \in E(T)$ on the line L, then all edges on the line segment L belong to $E(T)$.*

Let C' be a minimum-weight cycle satisfying properties (1), (2), (3), and (4), and let w' be a vertex of degree 4 in the internal region of C'. Let $E_{w'}$ be the set of edges of T not in the outer region of C', and each edge either belongs to the horizontal line or vertical line containing w'.

Claim 2. *Graph $G' = C' \cup E_{w'}$ is a subgrid of G. Moreover, $V(G') \subseteq V(G) \setminus U$, $E(G') \subseteq E(T)$, and all subdivision vertices in G' have degree exactly 2 in T.*

Proof. The definition of G' implies that $V(G') \subseteq V(G) \setminus U$ and $E(G') \subseteq E(T)$. Let L_1 and L_2 be the horizontal and vertical line passing through the point $w' = (w'_x, w'_y)$, respectively. We

show that G' is indeed a subgrid of G. Let l, r be degree 4 vertices of T on the line L_1 such that $l_x < w'_x, r_x > w'_x$, and the distances recdist(w', l) and recdist(w', r) are minimized. Similarly, let a, b be degree 4 vertices of T on the line L_2 such that $a_y > w'_y, b_y < w'_y$, and the distances recdist(w', a) and recdist(w', b) are minimized. Let $R = \{a_y, w'_y, b_y\}$ and $C = \{l_x, w'_x, r_x\}$. Now we will show that the subgrid G'', of G, defined by R and C is same as G'. Let L'_1 be the line segment of L_1, between l and r. This line segment is not in the external region of C'. Similarly, the line segment L'_2 on L_2, between a and b, is not in the external region of C'.

Let C'' be the cycle formed by the boundary vertices of G''. We need to show that C'' is the same as C'. We first show the following:

- *Condition (a)*: There is no edge $uv \in E(T)$ such that uv is in the internal region of C'' and $uv \in E(T) \setminus E(G'')$.

Suppose not. Among all such edges, let uv be an edge such that recdist(w', u) is minimized in the Hanan grid G. As uv does not belong to $E(G'')$, it does not lie on the line segments L'_1 and L'_2. Since uv is an internal edge of C'', $l_x < u_x < r_x$ and $b_y < u_y < a_y$. Notice that any shortest path, in G, between u and w' lies in the internal region of C''. In other words, the grid G_u defined by the u and w' lies in the internal region of C''. Any edge in G_u has shorter distance to w' than uv. Thus, since uv has minimum distance to w', if an edge, of G_u, does not belong to L'_1 or L'_2, then the edge cannot belong to $E(T)$. We show that uv is not in the external region of C'. Suppose uv is in the external region of C'. Since, w' is in the internal region of C', a shortest path from u to w' must cross into the internal region bounded by C'. As L'_1 and L'_2 are also not in the external region of C', there is an edge of $E(G_u) \setminus (L'_1 \cup L'_2)$, that belongs to C'. Therefore, there is an edge in G_u that belongs to T. This edge belongs to $E(T) \setminus E(G'')$ and is in the internal region of C'' but is closer to w' than uv. This contradicts the choice of uv. Thus, uv is not in the external region of C'. Consider the line L passing through uv. As mentioned earlier, $L \notin \{L_1, L_2\}$. Then L hits the line L_i, where L_i is exactly one of L_1 or L_2, at a single point $\tilde{u}$. First, suppose $u \neq \tilde{u}$. Let the line segment L' of L connect u and $\tilde{u}$. As shown earlier, u either belongs to C' or is in the internal region of C'. Following from Observation 6, since uv is an edge of T, there is another edge ux incident with u and lying on the line segment L'. This edge is also in the internal region of C'' and does not belong to $E(G'')$. Consider the other endpoint x. This vertex has shorter distance to w' than u. This contradicts the fact that uv was the chosen edge.

Finally, if $u = \tilde{u}$, then u lies on the line L_i. The line segments of L'_1 and L'_2 are not in the external region of C'. Note that $l_x < u_x < r_x$ and $b_y < u_y < a_y$. This means that the edge uv and the two edges of L_i, incident on u, belong to T and are not in the external region of C'. Hence, there are both horizontal and vertical edges in T that are incident with u. Since u is not an external vertex of C', the degree of u in T is 4 (by Observation 6). However, u is a vertex on L_i, $l_x < u_x < r_x$, and $b_y < u_y < a_y$. This contradicts the choice of one of l, r, a, or b.

Condition (a) implies that the internal region of C'' does not have an edge of T. Since all vertices $\{\ell, r, a, b, w'\}$ either belong to C' or to the internal region of C', the internal region of C'' is a subset of the internal region of C'. In addition, all edges in C'' are not in the outer region of C'. Since degree of l, r, a, and b in T are 4, by Claim 1, all edges in C'' belong to $E(T)$. Since w' is in the internal region of C'', property (4) holds for C''. As well, the vertices and edges of C'' either belong to C' or are in the internal region of C'. Thus, properties (1) and (2) also hold. Then, by the minimality of C', $C'' = C'$. Since the degree of w' in T is 4, by Claim 1, all edges in $E_{w'}$ belong to $E(T)$.

Now, we need to show that all subdivision vertices of G' have degree 2 in T. Suppose not. Let u be a subdivision vertex in G' such that degree of u in T is greater than 2. By Observation 6, degree of u in T is 4. This implies that there is an edge uv in the internal region of C' and $uv \notin E(G')$. This

contradicts condition (a). We have shown that $G' = C' \cup E_{w'}$ is a subgrid, where all subdivision vertices are of degree 2 in T. □

The next claim provides us with the insight on how subpaths of $\widehat{T}$ and T_{opt} behave in G'.

Claim 3. *Let F_h and F_v be the sets of horizontal and vertical edges in $G' = C' \cup E_{w'}$, respectively. Then exactly one of the following conditions is true:*

(1) $F_h \subseteq E(\widehat{T})$ and $F_v \subseteq E(T_{\text{opt}})$.
(2) $F_v \subseteq E(\widehat{T})$ and $F_h \subseteq E(T_{\text{opt}})$.

Proof. Let $G_1 = G'[E(\widehat{T})]$ and $G_2 = G'[E(T_{\text{opt}})]$. First, we show that each component of G_1 and G_2 is a path where all edges are of the same kind (i.e., either all are horizontal or all are vertical). Note that a component of G_1 or G_2, with only horizontal or only vertical edges, must be a path. For contradiction's sake, suppose there is a component with both horizontal and vertical edges. Without loss of generality, we assume that there is a component $C_1 \in G_1$ with both kinds of edges. This implies that there is a vertex $v \in V(C)$ such that v is incident with a vertical edge and a horizontal edge. However, by Observation 6(i) and the definition of G', such a vertex cannot be in G'. Therefore, each component of G_1 and G_2 is a path where all edges are of the same kind.

Next, we show that the edges of G_1 are either all horizontal or all vertical. For ease of notation, we will call a set of edges, which are all horizontal or all vertical, to be *parallel* edges. Let D be a component of G_1 and e an edge of $E(G_1) \setminus E(D)$. In addition, the edges of D are of a different orientation than the edge e. In other words, if all edges of D are horizontal, then e is a vertical edge and vice versa. As shown earlier, all edges of D are parallel to each other. Among all such pairs (D, e), we choose a pair (C_1, uv) that has the minimum distance between D and e in G'. As G' is connected, there is a path between C_1 and uv. Without loss of generality, assume that all edges of C_1 are horizontal and uv is a vertical edge. Assume that u and a vertex $w \in C_1$ are the vertices whose shortest path Q_{uw} in G' is a witness to the vertical edge uv and the component C_1 having the minimum distance between them. We first show that no edge of Q_{uw} belongs to $E(\widehat{T})$. Traversing from u along Q_{uw}, let e^* be the first edge of $E(\widehat{T})$ that is encountered. If e^* is a vertical edge, then (C_1, e^*) is a pair that satisfies the preceding description. However, the distance between C_1 and e^* is strictly smaller than the distance between C_1 and uv, which is a contradiction. But suppose e^* is a horizontal edge and it belongs to the component $C_2 \neq C_1$ of G_1. Then (C_2, uv) is a closer pair than (C_1, uv). Thus, all edges of Q_{uw} belong to $E(T_{\text{opt}})$ and not to $E(\widehat{T})$. Moreover, all edges of Q_{uw} must belong to a single component C_2 of G_2. This implies that all edges of Q_{uw} are parallel to each other. Let e_w be the edge of Q_{uw} that is incident with w. Since w has at least one horizontal edge of $E(C_1) \subseteq E(\widehat{T})$ and $e_w \notin E(\widehat{T})$, by Observation 6(i) with respect to w, e_w must be a vertical edge. Since, C_2 is a component of G_2, all edges of Q_{uw} must be vertical edges. Let e_u be the edge of Q_{uw} incident with u. Both $e_u \in G_2$ and $e \in G_1$ are vertical edges, where $e \in E(\widehat{T})$ while $e_u \notin E(\widehat{T})$. This is a contradiction to Observation 6(i) with respect to u. A similar argument shows that $E(G_2)$ is a set of parallel edges. This completes the proof of Claim 3. □

Note that G' is a 3×3 subgrid of G. Let G' be formed by horizontal paths $P_{123}, P_{456}, P_{789}$ and vertical paths $P_{147}, P_{258}, P_{369}$. Let $u_1, \ldots, u_9$ be the nine vertices in G' such that the path P_{ijk}, where $i, j, k \in [9]$, contains the vertices u_i, u_j, and u_k. Due to Claim 3, without loss of generality, we may assume that the horizontal paths belong to T_{opt} and the vertical paths belong to $\widehat{T}$. For a path P_{ijk}, we use P_{ij} and P_{jk} to denote the subpaths of P_{ijk} connecting u_i and u_j, and u_j and u_k, respectively. Let the length of the subpath P_{12} be ℓ_1 and the length of the subpath P_{23} be ℓ_2. By the definition of

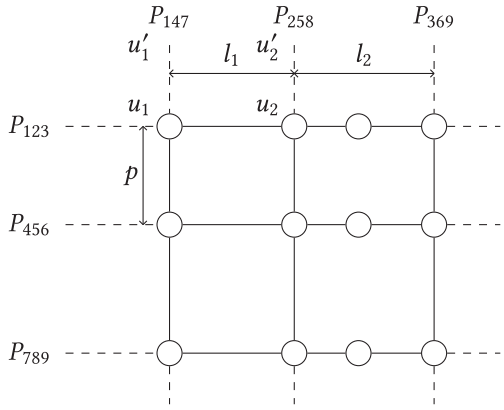

Fig. 4. The subgrid G'.

G', the length of P_{45} is also ℓ_1, and the length of P_{56} is ℓ_2. In addition, let the length of the subpath P_{14} be p. See Figure 4.

Suppose $\ell_1 + \ell_2 > 2p$. Then, we consider the graph T^* formed by deleting in T_{opt} the path P_{123} and adding the two paths P_{14} and P_{36}. Since all subdivision vertices of G' are of degree 2 in T, T^* is a Steiner tree of weight strictly less than the weight of T_{opt}. This contradicts the choice of T_{opt}. Hence, this is not possible.

Suppose $\ell_1 + \ell_2 \leq 2p$. Without loss of generality, let $\ell_1 \leq \ell_2$. Thus, $\ell_1 \leq p$. Consider the two paths P_{147} and P_{258}. They are vertical paths of $\widehat{T}$ such that all vertices in these paths belong to $V(G) \setminus U$ (i.e., non-terminals and non-bend vertices) and have degree 2 in $\widehat{T}$ (by Observation 6). This implies that all edges in any path $R \in \{P_{147}, P_{258}\}$ are added in a single step while constructing $\widehat{T}$. Since both the paths are parallel, by Observation 4, if a path R_1 in $\widehat{T}$ has both P_{147} and P_{258} as subpaths, then R cannot be a shortest path between its endpoints. Thus, by construction of $\widehat{T}$, both P_{147} and P_{258} could not have been added to $\widehat{T}$ in a single step of the construction. Also by construction, one of them is added to $\widehat{T}$ before the other. Without loss of generality, let P_{147} be added before P_{258}. Again by construction, a path, containing P_{258} as a subpath, was added in the i-th step to connect a terminal z to the already constructed $\widehat{T}_{i-1}$. Let R^* be the path added to T_{i-1}. By definition of G', this terminal z must lie outside the region formed by the subgrid G'. Since P_{258} was part of a shortest path between $\widehat{T}_{i-1}$ and z, by Observation 4, z must lie on a row strictly higher than or strictly lower than the rows in G'. Suppose this terminal lies above P_{123}. By Observation 6, both vertical edges incident on u_1 belong to $\widehat{T}$. Since u_1 and u_2 are of degree 2 in $\widehat{T}$, both vertical edges incident with u_1, as well as u_2, are added when the path containing P_{147} or P_{258} is added to construct $\widehat{T}$. This implies that both vertical edges, incident with u_1, are present in T_{i-1}. Let $u_1u'_1$ be the vertical edge present in T_{i-1} and not in $E(P_{147})$. Let $u_2u'_2$ be the vertical edge not present in P_{258}. Now, consider the path R_2, between u'_2 and z, obtained by concatenating the horizontal path between u'_1 and u'_2 and the subpath of R^* connecting u'_2 and z. The length of the path R_2 is strictly less than that of R^* and $u'_1 \in T_{i-1}$. This is a contradiction. The case when z lies below any row in G' is identical to the preceding case.

Thus, there is no such subgrid G' of G such that G' is a subgraph of T. This implies that there is no $9\sqrt{n} \times 9\sqrt{n}$ grid as a minor in T. Due to Proposition 2.7, the treewidth of T must be at most $\frac{9}{2} \cdot 9\sqrt{n} \leq 41\sqrt{n}$. This completes the proof.

3.3 Dynamic Programming Algorithm for Rectilinear Steiner Tree

In this section, we utilize all of the results proved in the previous sections and design our algorithm for Rectilinear Steiner Tree. By Lemma 3.4, we known that given a shortest path RST $\widehat{T}$, there exists an optimum Steiner tree T_{opt} such that the treewidth of $T = \widehat{T} \cup T_{\text{opt}}$ is at most $41\sqrt{n}$. The idea of the algorithm is to *implicitly* do a dynamic programming over a tree decomposition of T, *even though we do not know what T is*, to compute an optimum Steiner tree for T.

Suppose we know the subgraph T of G such that there is an optimum Steiner tree fully contained in T. Then we can do the well-known algorithm for Steiner Tree over the tree decomposition of T (see Theorem 7.8 in Cygan et al. [3]). However, in our case, the difficulty is that we do not know T and therefore do not have a nice tree decomposition of T. Let $(\mathbb{T}, \mathcal{X}' = \{X'_t\}_{t \in V(\mathbb{T})})$ be a potential nice tree decomposition of T, of width at most $41\sqrt{n}$, where $\mathbb{T}$ is rooted tree with root r. To describe our algorithm in the best way possible, we modify our tree decomposition slightly. First, we fix a terminal $z^* \in K$. From $(\mathbb{T}, \mathcal{X}' = \{X'_t\}_{t \in V(\mathbb{T})})$, we obtain a new tree decomposition $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ by adding z^* to each bag X'_t, $t \in V(\mathbb{T})$. We continue to name a bag X_t as we named X'_t—that is, if X'_t was a leaf bag, then so is X_t and so on. Notice that the treewidth of $(\mathbb{T}, \mathcal{X})$ increases by at most 1, but the root and leaf bags of $\mathbb{T}$ are identical to the singleton set $\{z^*\}$. For a node t, let V_t be the union of all bags present in the subtree of $\mathbb{T}$ rooted at t. For a node t, we define a graph $S_t = (V_t, E_t = \{e \in E(T) \ : \ e \text{ is introduced in the subtree rooted at } t\})$. We define the following function $c[\cdot]$ for the Steiner Tree problem: for each bag X_t, $X \subseteq X_t$ and a partition $\mathcal{P} = (P_1, \ldots, P_q)$ of X with $P_i \neq \emptyset$, the value $c[t, X, \mathcal{P}]$ is the minimum weight of a subgraph F of S_t with the following properties:

(1) F has exactly q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_i = X_t \cap V(C_i)$ for all $i \in [q]$. In other words, $\mathcal{P}$ corresponds to connected components of F.
(2) $X_t \cap V(F) = X$. In other words, the vertices of $X_t \setminus X$ are untouched by F.
(3) $K \cap V_t \subseteq V(F)$. In other words, all terminal vertices in S_t belong to F.

Note that if we consider the input graph T with tree decomposition $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ and restrict an optimal Steiner tree to the subgraph S_t, then the restriction F will be a forest $C_1, \ldots, C_q$. Not all vertices in X_t need to be in F, and we may denote the vertices of X_t that are in the restriction as X. From the C_i's, we may obtain a partition $\mathcal{P} = (P_1, \ldots, P_q)$ of X such that $P_i = X_t \cap V(C_i)$. By the properties of tree decomposition, all terminals in V_t must belong to F, as otherwise it will contradict the fact that F is a restriction of an optimal Steiner tree. Thus, the properties we have listed previously help to describe an optimal Steiner tree when restricted to a subgraph S_t. Thus, the value $c[t, X, \mathcal{P}]$ stores the length of this subgraph F of the optimal Steiner tree. In particular, by definition of the function $c[\cdot]$, for the root r of $\mathbb{T}$ the value $c[r, \{z^*\}, \{\{z^*\}\}]$ corresponds to the weight of a minimum Steiner tree. Thus, knowing the function $c[\cdot]$ is enough to know the weight of an optimal rectilinear Steiner tree of K in T.

In our case, we do not know the graph $T = \widehat{T} \cup T_{\text{opt}}$ and a tree decomposition of T, but we know that the treewidth of T is at most $41\sqrt{n}$. This implies that the number of choices for bags in a tree decomposition of T is at most $n^{O(\sqrt{n})}$. Consider properties (1) and (2) mentioned earlier. They are *local* properties of the witness subgraph F with respect to the bag X_t. However, property (3) states that all terminals in the subgraph S_t should be present in F. Moreover, since $\widehat{T}$ is a rectilinear Steiner tree, all terminals in S_t are definitely covered by $\widehat{T}$. In fact, we can bound the *potential* sets $K \cap V(S_t)$ using $\widehat{T}$. Observe that any bag X_t is a separator of size $O(\sqrt{n})$ for T and thus for $\widehat{T}$. This implies that *for every connected component C of $\widehat{T} - X_t$, either $K \cap V(C)$ is fully in V_t or no vertex in $K \cap V(C)$ belongs to V_t*. Since each vertex in G has degree at most 4 and X_t is a bag in a tree decomposition, the number of connected components in $\widehat{T} - X_t$ is at most $4|X_t| \leq 164(\sqrt{n} + 1)$.

As observed before, for any connected component C of $\widehat{T} - X_t$, either $K \cap V(C)$ is fully in V_t or no vertex in $K \cap V(C)$ belongs to V_t. This implies that the number of potential sets $K' \subseteq K$ such that $K' = (V_t \setminus X_t) \cap K$ is at most $2^{164(\sqrt{n}+1)}$, and we can enumerate them in subexponential time. Using this observation, we could keep track of property (3) as well, even though we do not know the graph T and its tree decomposition.

Now, we move toward the formal explanation of our algorithm. We first define the notion of a *type*, which is the analogue of a tuple $(t, X, \mathcal{P})$ that is an input for the function $c[\cdot]$ defined earlier.

Definition 3.5. A type is a tuple $(Y, Y' \subseteq Y, \mathcal{P}, K')$ such that the following hold:

(i) Y is a subset of $V(G)$ of size at most $41\sqrt{n} + 2$.
(ii) $\mathcal{P}$ is a partition of Y' with q blocks.
(iii) There exists a set of components $C_1, \ldots, C_q$ in $\widehat{T} - Y$ such that $K' = K \cap (Y' \cup \bigcup_{i=1}^{q} V(C_i))$.

Informally, in a type $(Y, Y', \mathcal{P}, K')$, Y represents a potential bag Y of a node (say t) in a tree decomposition of T. The set Y' and partition $\mathcal{P}$ have the same meaning as that of $(t, Y', \mathcal{P})$ as an input to the function $c[\cdot]$. The set K' is the set of terminals in the graph S_t. The next lemma gives an upper bound on the number of types.

Lemma 3.6. *There is a $2^{O(\sqrt{n}\log n)}n^{O(1)}$ time algorithm $\mathcal{B}$ enumerating all of the types.*

Proof. We know that a type is a tuple $(Y, Y', \mathcal{P}, K')$ satisfying properties (i) through (iii) of Definition 3.5. Since $|V(G)| \leq n^2$, the number of choices for Y is $n^{O(\sqrt{n})}$. The cardinality of Y is at most $41\sqrt{n} + 2$, and thus for a fixed Y, the number of choices for Y' is $2^{O(\sqrt{n})}$ and the number of choices for the partition $\mathcal{P}$, of Y', is $n^{O(\sqrt{n})}$. By definition of the Hanan grid G, each vertex in G has at most four neighbors. Thus, $\widehat{T} - Y$ has at most $4(41\sqrt{n} + 2)$ components. Hence, on fixing Y, the choices for K' is at most $2^{O(\sqrt{n})}$. Thus, we get an upper bound of $2^{O(\sqrt{n}\log n)}$ on the number of types. Furthermore, it can be enumerated in time $2^{O(\sqrt{n}\log n)}n^{O(1)}$. □

Our algorithm is a dynamic programming algorithm over the types of T. Let $N = 3(|V(G)| + |E(G)|)$. Our algorithm computes values $\mathcal{A}[i, D]$, where $i \in [N]$ and D is a type. We want the table $\mathcal{A}[,]$ to contain all of the information that is necessary for correctly computing the function $c[\cdot]$ for Steiner Tree over a tree decomposition of T. To motivate the definition of $\mathcal{A}[,]$, we assume a *hypothetical tree decomposition* $\mathcal{T} = (\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ of T of width at most $41\sqrt{n}$. For ease of understanding, let it be a nice tree decomposition, and let the tree be rooted at a node $r \in \mathbb{T}$. Recall that the level of a vertex $t \in \mathbb{T}$ is the height of the subtree of $\mathbb{T}$ rooted at t. The *height* of a node t is the number of vertices in the longest downward path to a leaf from that node. Following from Proposition 2.1, note that the level of any node of $\mathbb{T}$ is at most N. Suppose $t \in \mathbb{T}$ is a node at level i and corresponds to the bag X_t. Let V_t be the union of bags present in the subtree rooted at t. Let the graph S_t be defined as $(V_t, \{e \mid e \text{ is introduced in the subtree rooted at } t\})$. Let $K' = V_t \cap K$. Then, for any $X \subseteq X_t$, and a partition $\mathcal{P}$ of X having q blocks, $\mathcal{A}[i, (X_t, X, \mathcal{P}, K')] = c[t, X, \mathcal{P}]$. As mentioned before, $c[t, X, \mathcal{P}]$ is the minimum weight of the subgraph F of S_t such that the following hold: (i) F has q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_j = X_t \cap V(C_j)$, (ii) $X_t \cap V(F) = X$, and (iii) $K \cap V_t \subseteq V(F)$. For other pairs (i, D), we do not guarantee that the value of $\mathcal{A}[i, D]$ is meaningful. However, it is enough to maintain reasonable values for only the preceding subset of pairs (i, D). Of course, we do not know T, and thus we do not know the tree decomposition $\mathcal{T}$. Therefore, we store values in the table $\mathcal{A}[,]$ in such a way that given *any* nice tree decomposition of T, we have information pertaining to it. Thus, given a pair (i, D) where $D = (Y, Y' \subseteq Y, \mathcal{P}, K')$, we view Y as a bag of some hypothetical nice tree decomposition, $\mathcal{T}$, of T and assume that the level

of the bag corresponding to Y in $\mathcal{T}$ is i. At a level i of this hypothetical nice tree decomposition, any bag is one of at most five kinds, namely the five kinds of bags as per the definition of a nice tree decomposition. We guess the relationship between a bag at level i and its children, which must be at level at most $i-1$. For example, if our hypothetical node t corresponds to an introduce vertex bag X_t, then we pretend that we know X_t, the child node t', the bag $X_{t'}$, and the vertex v that is being introduced at node t. Thereafter, for a subset $X \subseteq X_t$, and a partition $\mathcal{P}$ of X, we try to compute $\mathcal{A}[i,(X_t,X,\mathcal{P},K')]$ using that values of $\mathcal{A}$ calculated at step $i-1$ of the algorithm. The calculation ensures that $\mathcal{A}[i,(X_t,X,\mathcal{P},K')] = c[t,X,\mathcal{P}]$. In what follows, we give a formal definition of $\mathcal{A}[,]$.

We write a recurrence relation for $\mathcal{A}[i,D]$, where $i \in [N]$ and D is a type. We fix a terminal z^* in K. When $i = 1$,

$$\mathcal{A}[1,D] = \begin{cases} 0 & \text{if } D = (\{z^*\},\{z^*\},\{\{z^*\}\},\{z^*\}) \\ \infty & \text{otherwise.} \end{cases} \tag{1}$$

To define $\mathcal{A}[i,D]$ for $i \in [N] \setminus \{1\}$ and a type $D = (Y,Y',\mathcal{P},K')$, we first define many intermediate values and take the minimum over all such values.

We first try to *view* Y as an introduce node in some nice tree decomposition of T and having level i. This viewpoint results in the following recurrence. For all $v \in Y$ where type $D = (Y,Y',\mathcal{P},K')$,

$$I_v[i,D] = \begin{cases} \infty & \text{if } v \notin Y' \text{ and } v \in K \\ \mathcal{A}[i-1,(Y\setminus\{v\},Y',\mathcal{P},K')] & \text{if } v \notin Y' \text{ and } v \notin K \\ \mathcal{A}[i-1,(Y\setminus\{v\},Y'\setminus\{v\},\mathcal{P}\setminus\{\{v\}\},K'\setminus\{v\})] & \text{if } v \in Y' \end{cases} \tag{2}$$

Intuitively, if Y is a bag corresponding to a node t in a tree decomposition of T and K' is the set of terminals in S_t, then Equation (2) corresponds to the value of the function $c[t,Y',\mathcal{P}]$ for Steiner Tree.

For all $u,v \in Y$ and $uv \in E(G)$ where type $D = (Y,Y',\mathcal{P},K')$,

$$I_{uv}[i,D] = \min\left\{\min_{\mathcal{P}'}\left\{\mathcal{A}[i-1,(Y,Y',\mathcal{P}',K')] + \mathrm{recdist}(uv)\right\}, \mathcal{A}[i-1,D]\right\}, \tag{3}$$

where $\mathcal{P}'$ varies over partitions of Y' such that u and v are in different blocks of $\mathcal{P}'$ and by merging these two blocks we get the partition $\mathcal{P}$. Note that if $\{u,v\} \not\subseteq Y'$ or u and v are in same block of $\mathcal{P}$, then Equation (3) gives $I_{uv}[i,D] = \mathcal{A}[i-1,D]$. Equation (3) corresponds to the computation of values in the introduce edge node where the edge uv is introduced.

For all $w \in V(G)$,

$$F_w[i,D] = \min\left\{\min_{\mathcal{P}'}\left\{\mathcal{A}[i-1,(Y\cup\{w\},Y'\cup\{w\},\mathcal{P}',K')]\right\}, \mathcal{A}[i-1,(Y\cup\{w\},Y',\mathcal{P},K')]\right\}, \tag{4}$$

where $\mathcal{P}'$ in the inner minimum varies over all partitions obtained by adding w to one of the existing blocks. Equation (4) corresponds to computation in a forget node where w is forgotten.

$$J[i,D] = \min_{\substack{\mathcal{P}=\mathcal{P}_1\sqcup\mathcal{P}_2 \\ K'=K'_1\cup K'_2 \\ i'\le i-1}} \{\mathcal{A}[i-1,(Y,Y',\mathcal{P}_1,K'_1)] + \mathcal{A}[i',(Y,Y',\mathcal{P}_2,K'_2)]\} \tag{5}$$

Equation (5) corresponds to a computation in a join node.

We define $\mathcal{A}[i,D]$ for $i \in [N] \setminus \{1\}$ and type $D = (Y,Y',\mathcal{P},K')$ as follows.

$$\mathcal{A}[i,D] = \min\begin{cases} \min_{v\in Y} I_v[i,D] \\ \min_{\substack{uv\in E(G) \\ u,v\in Y}} I_{uv}[i,D] \\ \min_{w\in V(G)} F_w[i,D] \\ J[i,D] \end{cases} \tag{6}$$

For each $i \in [N]$ and each type D, we associate with $A[i, D]$ a subgraph of T. We say that a subgraph F is of type $(Y, Y', \mathcal{P}, K')$, where $\mathcal{P} = \{P_1, \dots P_q\}$ if the following holds:

(a) The number of connected components in F is equal to $|\mathcal{P}| = q$. We can order the connected components $C_1, \dots, C_q$ of F such that $V(C_j) \cap Y = P_j$.
(b) $V(F) \cap K = K'$.

In the following lemma, we show the connection between $A[i, D]$ and a subgraph of type D.

Lemma 3.7. *Let $i \in [N]$ and D be a type. Furthermore, let $A[i, D]$ be computed by the Equation (6), and let $A[i, D] = \ell$. Then there is a subgraph F, of type D, such that* $\text{recdist}(F) \leq \ell$.

Proof. We prove the statement using induction on i. Recall that we fixed a terminal z^* in K. Since the graph $(\{z^*\}, \emptyset)$ is of type $(\{z^*\}, \{z^*\}, \{\{z^*\}\}, \{z^*\})$, the base case holds trivially. Let $1 < i \leq N$ and $D = (Y, Y', \mathcal{P}, K')$ be a type and $\mathcal{A}[i, D] = \ell$. We need to show that there is a subgraph F of G such that F has type D and $\text{recdist}(F) \leq \ell$. We know that $\mathcal{A}[i, D]$ is computed using Equation (6), which is a minimum over a set of values. Suppose $\mathcal{A}[i, D] = I_v[i, D] = \ell$ for some $v \in Y$. If $v \notin Y'$ and $v \notin K$, then by Equation (2), $\ell = I_v[i, D] = \mathcal{A}[i - 1, (Y \setminus \{v\}, Y', \mathcal{P}', K')]$. By the induction hypothesis, there is a subgraph F of type $D' = (Y \setminus \{v\}, Y', \mathcal{P}', K')$ and $\text{recdist}(F) \leq \ell$. Since D and D' are satisfying types, conditions (a) and (b) in the definition of satisfying types imply that F is of type D as well. If $v \in Y'$, then $\ell = I_v[i, D] = \mathcal{A}[i - 1, D'' = (Y \setminus \{v\}, Y' \setminus \{v\}, \mathcal{P} \setminus \{v\}, K' \setminus \{v\})]$. By the induction hypothesis, there is a subgraph F such that $\text{recdist}(F) \leq \ell$ and F is of type D''. This implies that the graph $F' = F \cup (\{v\}, \emptyset)$ is of type D and $\text{recdist}(F') = \text{recdist}(F) \leq \ell$.

Next, suppose $\mathcal{A}[i, D] = J[i, D] = \ell$. Suppose the type $D = (Y, Y', \mathcal{P}, K')$. Then by Equation (5), there are partitions $\mathcal{P}_1$ and $\mathcal{P}_2$ with $\mathcal{P} = \mathcal{P}_1 \sqcup \mathcal{P}_2$, terminal subsets K'_1 and K'_2 such that $K'_1 \cup K'_2 = K'$, and an $i' \leq i - 1$ such that $\ell = J[i, D] = \mathcal{A}[i - 1, (Y, Y', \mathcal{P}_1, K'_1)] + \mathcal{A}[i', (Y, Y', \mathcal{P}_2, K'_2)]$. By the induction hypothesis, there is a subgraph F_1 of type $D_1 = (Y, Y', \mathcal{P}_1, K'_1)$ and a subgraph F_2 of type $D_2 = (Y, Y', \mathcal{P}_2, K'_2)$ such that $\text{recdist}(F_1) + \text{recdist}(F_2) \leq \ell$. Consider the union graph $F = F_1 \cup F_2$. By definition, $\text{recdist}(F) \leq \text{recdist}(F_1) + \text{recdist}(F_2) \leq \ell$. We show that F is of type D and then we will be done. To show this, we need to prove conditions (a) and (b) in the definition of type of a subgraph. First, we show that the number of components in F is equal to $|\mathcal{P}|$. Consider a block $P \in \mathcal{P}$. Let $\{P^1, P^2, \dots, P^a\}$ be the blocks of $\mathcal{P}_1$ that are contained inside P. Similarly, let $\{P'^1, P'^2, \dots, P'^b\}$ be the blocks of $\mathcal{P}_2$ that are contained inside P. Consider the auxiliary graph G_{aux} where the vertices correspond to $\{P^1, P^2, \dots, P^a\} \cup \{P'^1, P'^2, \dots, P'^b\}$ and there is an edge between two blocks if there is a vertex of G that belongs to both blocks. By definition of a partition join, the graph G_{aux} must be connected. Now, by the induction hypothesis, $\{P^1, P^2, \dots, P^a\}$ correspond to components $\{C^1, C^2, \dots, C^a\}$ in F_1. Similarly, $\{P'^1, P'^2, \dots, P'^b\}$ correspond to components $\{C'^1, C'^2, \dots, C'^b\}$ in F_2. Since G_{aux} is a connected graph, $\bigcup_{1 \leq j \leq a} C^j \cup \bigcup_{1 \leq k \leq b} C'^k$ must also be a connected graph. Thus, the block P of $\mathcal{P}$ corresponds to a component of F. Similarly, consider a component C of F. Let $\{C^1, C^2, \dots, C^a\}$ be the components of $F_1 \cap C$ and $\{C'^1, C'^2, \dots, C'^b\}$ be the components of $F_2 \cap C$. By the induction hypothesis, these components correspond to blocks $\{P^1, P^2, \dots, P^a\}$ in $\mathcal{P}_1$ and $\{P'^1, P'^2, \dots, P'^b\}$ in $\mathcal{P}_2$. Since C is connected, the graph G_{aux} is also connected. This means that all of these blocks must be in the same block P of $\mathcal{P}$. Thus, a component of F corresponds bijectively to a block $P \in \mathcal{P}$. We can order the components such that the ordering matches the ordering of the corresponding blocks in $\mathcal{P}$. Second, the subgraph F_1 contained the terminals of K'_1 and the subgraph F_2 contained the terminals of K'_2. Thus, by definition, the subgraph F contains the terminal set $K' = K'_1 \cup K'_2$. Therefore, the subgraph F is of type D and $\text{recdist}(F) \leq \ell$.

In all other cases, the proof follows by similar arguments. □

The next lemma helps us relate an optimal rectilinear Steiner tree to the values computed for the table $\mathcal{A}[,]$. First we recall the definition of $c[,,]$. For a subset $X \subseteq X_t$, and a partition $\mathcal{P}$ of X with q

blocks, let $c[t, X, \mathcal{P}]$ be the minimum weight of the subgraph F of S_t such that the following hold: (i) F has q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_j = X_t \cap V(C_j)$, (ii) $X_t \cap V(F) = X$, and (iii) $K \cap V_t \subseteq V(F)$. If there is no such subgraph F, then the value of $c[t, X, \mathcal{P}]$ is ∞.

Lemma 3.8. *Let $\mathcal{T} = (\mathbb{T}, \{X_t\}_{t \in V(\mathbb{T})})$ be a nice tree decomposition of $T = \widehat{T} \cup T_{\text{opt}}$. For a node t, let X_t be the corresponding bag, $X \subseteq X_t$, $\mathcal{P}$ be a partition of X, V_t be the union of bags in the subtree rooted at t, and $K' = K \cap V_t$. Then $A[i, (X_t, X, \mathcal{P}, K')] \leq c[t, X, \mathcal{P}]$.*

Proof. Recall that z^* is a fixed terminal. We add z^* to all bags in $\mathcal{T}$. This new decomposition still satisfies all properties of a tree decomposition. The width of this new tree decomposition increases by at most 1. This is no longer a nice tree decomposition as per the definition in Section 2. However, we will call a bag in the new tree decomposition with the same name as in the initial nice tree decomposition (e.g., leaf bag). For ease of presentation, we use $\mathcal{T} = (\mathbb{T}, \{X_t\}_{t \in V(\mathbb{T})})$ to denote the new tree decomposition of T. Note that all leaf bags and the root bag contain only one element, z^*. Let r be the root of $\mathbb{T}$. For any node $t \in V(\mathcal{T})$, we define the *level* of t as the height of the subtree rooted at t. Recall that the *height* of a node t is the number of vertices in the longest downward path to a leaf from that node. Note that leaves in $\mathcal{T}$ other than root r have level 1 and the level of r is the height of $\mathbb{T}$. By Proposition 2.1, the level of any node in $\mathbb{T}$ is at most N. For any node $t \in V(\mathcal{T})$, we use ℓ_t to denote the level of t. For any t, we denote the graph S_t as $(V_t, \{e \mid e \text{ is introduced in the subtree rooted at } t\})$, where V_t is the union of bags present in the subtree rooted at t.

We prove the following statement: for any $t \in V(\mathbb{T})$, $X \subseteq X_t$ and a partition $\mathcal{P}$ of X, it is true that $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, K \cap V_t)] \leq c[t, X, \mathcal{P}]$. We prove the statement using induction on the level of the node t. The base case is when $\ell_t = 1$. In this case, $X_t = \{z^*\}$. If $X = \{z^*\}$ and $\mathcal{P} = \{\{z^*\}\}$, by definition $\mathcal{A}[1, (X_t, X, \{X\}, K \cap V_t)] = 0 = c[t, X, \{X\}]$. Otherwise, $\mathcal{A}[1, (X_t, X, \mathcal{P}, K \cap V_t)] = \infty = c[t, X, \{X\}]$. Let t be a node in $V(\mathcal{T})$, $X \subseteq X_t$ and $\mathcal{P}$ be a partition of X such that $1 < \ell_t \leq N$. Let $K' = K \cap V_t$. If $(X_t \setminus X) \cap K \neq \emptyset$, then by definition $c[t, X, \mathcal{P}] = \infty$ and so the statement holds. Suppose $(X_t \setminus X) \cap K = \emptyset$. Since $\widehat{T}$ is a Steiner tree for K, $T \subseteq V(\widehat{T})$. Since X is a bag in the tree decomposition $\mathcal{T}$, all terminals in a connected component C of $\widehat{T} - X_t$ are either fully contained in V_t or none of the terminals in the component C are present in V_t. Thus, there are connected components $C_1, \ldots, C_{\hat{j}}$ of $\widehat{T} - X_t$ such that $K' = K \cap V_t = K \cap (X_t \cup \bigcup_{j=1}^{\hat{j}} V(C_j))$. This implies that $(X_t, X, \mathcal{P}, K')$ is a type. Let $\mathcal{P} = \{P_1, \ldots, P_q\}$ and F be a witness subgraph for the value $c[t, X, \mathcal{P}]$. In other words, $\text{recdist}(F) = c[t, X, \mathcal{P}]$ and the following conditions hold: (i) F has q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_j = X_t \cap V(C_j)$, (ii) $X_t \cap V(F) = X$, and (iii) $K \cap V_t \subseteq V(F)$. We analyze cases based on the nature of the node t.

Case 1: t is an introduce vertex node. Let t' be the child of t and $\{v\} = X_t \setminus X'_t$. Note that the level of t' is $\ell_t - 1$. If $v \notin V(F)$, then $c[t', X, \mathcal{P}] \leq \text{recdist}(F)$. By Equations (6) and (2), $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, K')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, K')]$. By the induction hypothesis, it holds that $\mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, K')] \leq c[t', X, \mathcal{P}] \leq \text{recdist}(F) = c[t, X, \mathcal{P}]$. If $v \in V(F)$, then v appears as an isolated vertex in F, because by definition of an introduce vertex bag, v is an isolated vertex in S_t. This implies that $c[t', X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}] \leq \text{recdist}(F)$. By Equations (6) and (2), $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, K')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, K')]$. By the induction hypothesis, $\mathcal{A}[\ell_t - 1, (X'_t, X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, K')] \leq c[t', X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}] \leq \text{recdist}(F \setminus \{v\}) = \text{recdist}(F) = c[t, X, \mathcal{P}]$.

Case 2: t is an introduce edge node. Let t be labeled with the edge uv and t' be the child of t. In other words, $\{u, v\} \subseteq X_{t'} = X_t$. Note that the level of t' is $\ell_t - 1$. If $uv \notin E(F)$, then $c[t', X, \mathcal{P}] \leq \text{recdist}(F)$. By Equations (6) and (3), we know that $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, K')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, K')]$. By the induction hypothesis, $\mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, K')] \leq c[t', X, \mathcal{P}] \leq \text{recdist}(F) = c[t, X, \mathcal{P}]$.

Suppose $uv \in E(F)$. Let $C'_1, \ldots, C'_{q'}$ are the connected components of $F - uv$. Consider the partition $\mathcal{P}'$ to be $\{V(C'_1) \cap X, \ldots, V(C'_{q'}) \cap X\}$. This implies that $c[t', X, \mathcal{P}'] \leq \text{recdist}(F \setminus \{uv\}) = \text{recdist}(F) - \text{recdist}(uv)$. By induction hypothesis, $\mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}', K')] \leq c[t', X, \mathcal{P}'] \leq \text{recdist}(F) - \text{recdist}(uv)$. The property of F implies that in the partition $\mathcal{P}'$, if we merge the blocks containing u and v, we get the partition $\mathcal{P}$. Thus, by Equations (6) and (3), $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, K')] \leq \mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}', K')] + \text{recdist}(uv) \leq \text{recdist}(F)$.

Case 3: t is a forget node. Let t' be the child of t and $\{w\} = X_{t'} \setminus X_t$. Note that the level of t' is $\ell_t - 1$. If $w \notin V(F)$, then $c[t', X, \mathcal{P}] \leq \text{recdist}(F)$. By the induction hypothesis, $\mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}, K')] \leq c[t', X, \mathcal{P}] \leq \text{recdist}(F)$. By Equations (6) and (4), $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, K')] \leq \mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}, K')] \leq \text{recdist}(F)$.

Suppose $w \in V(F)$. Let C_j be the component of F containing w. Note that $P_j = V(C_j) \cap X$. Let $\mathcal{P}'$ be a partition obtained from $\mathcal{P}$, by adding w to the block P_j. Then $\mathcal{P}'$ is a partition of $X \cup \{w\}$. This implies that $c[t', X \cup \{w\}, \mathcal{P}'] \leq \text{recdist}(F)$. By the induction hypothesis, $\mathcal{A}[\ell_{t'}, (X_{t'}, X \cup \{w\}, \mathcal{P}', K')] \leq c[t', X \cup \{w\}, \mathcal{P}'] \leq \text{recdist}(F)$. By Equations (6) and (4), $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, K')] \leq \mathcal{A}[\ell_t - 1, (X_{t'}, X \cup \{w\}, \mathcal{P}', K')] \leq \text{recdist}(F)$.

Case 4: t is a join node. Let t_1 and t_2 be the children of t. Here, $X_t = X_{t_1} = X_{t_2}$, and the level of $X_{t_j}, j \in \{1, 2\}$, is at most $\ell_t - 1$. Note that the level of one of the children must be exactly $\ell_t - 1$. Without loss of generality, we assume that the level of X_{t_1} is exactly $\ell_t - 1$, whereas the level of X_{t_2} is ℓ_{t_2}, which could be less than $\ell_t - 1$. Let F_1 be the graph with vertex set as $V(F) \cap V_{t_1}$ and edge set as $E(F) \cap E(S_{t_1})$. Let F_2 be the graph with vertex set as $V(F) \cap V_{t_2}$ and edge set as $E(F) \setminus E(F_1)$. Note that $F = F_1 \cup F_2$. Let $K'_1 = V(F_1) \cap K$ and $K'_2 = V(F_2) \cap K$. Since all connected components in $V(F)$ contain at least one vertex from X and X_t is a bag in the tree decomposition, all connected components in F_1 and F_2 contain at least one vertex from X. Let $C'_1, \ldots, C_{q'}$ be the connected components of F_1 and $C''_1, \ldots, C''_{q''}$ be the connected components in F_2. Let $\mathcal{P}_1 = \{X \cap V(C'_i), \ldots, X \cap V(C'_{q'})\}$ and $\mathcal{P}_2 = \{X \cap V(C''_i), \ldots, X \cap V(C''_{q''})\}$. Thus, $c[t_1, X, \mathcal{P}_1] \leq \text{recdist}(F_1)$ and $c[t_2, X, \mathcal{P}_2] \leq \text{recdist}(F_2)$. By the induction hypothesis, $\mathcal{A}[\ell_{t_j}, (X_{t_j}, X, \mathcal{P}_j, K'_j)] \leq c[t_j, X, \mathcal{P}_j]$ for $j \in \{1, 2\}$. The definitions of F, F_1 and F_2 imply that $\mathcal{P} = \mathcal{P}_1 \sqcup \mathcal{P}_2$. By Equations (5) and (6), it follows that $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, K')] \leq \mathcal{A}[\ell_t - 1, (X_t, X, \mathcal{P}_1, K'_1)] + \mathcal{A}[\ell_{t_2}, (X_t, X, \mathcal{P}_2, K'_2)] \leq \text{recdist}(F_1) + \text{recdist}(F_2) = \text{recdist}(F)$. □

Finally, we describe the subexponential algorithm for Rectilinear Steiner Tree.

Theorem 3.9. Rectilinear Steiner Tree *has a deterministic algorithm with running time* $2^{O(\sqrt{n} \log n)} n^{O(1)}$.

Proof. We take as input a set K of n terminal points, the Hanan grid G of K, and the weight function recdist. Then, using Lemma 3.1, we compute a shortest path RST $\widehat{T}$. By Lemma 3.4, we know that there is an optimal Steiner tree T_{opt} with $\text{tw}(\widehat{T} \cup T_{\text{opt}}) \leq 41\sqrt{n}$. Based on the shortest path RST $\widehat{T}$, we apply Lemma 3.6 to enumerate all possible types D of G. We fix an integer $N = 3(|V(G)| + |E(G)|)$ and a terminal z^* in K. For each $i \in [N]$ and each type D, the algorithm computes values $\mathcal{A}[i, D]$, according to Equations (1) and (6). The values in $\mathcal{A}[,]$ are filled in the increasing order of i. Finally, the algorithm outputs $\min_{i \in [N]} \mathcal{A}[i, (\{z^*\}, \{z^*\}, \{\{z^*\}\}, T)]$.

For the hypothetical subgraph T, fix an optimal nice tree decomposition $\mathcal{T}$, rooted at node r. Add a fixed terminal z^* to each bag of the nice tree decomposition (as described in the proof of Lemma 3.8). The treewidth of this new tree decomposition is at most $41\sqrt{n} + 1$. In addition, by Proposition 2.1, the height of the tree decomposition is at most N. Let t be a node in the tree decomposition, of level ℓ_t. Let X_t be the bag of t and V_t be the union of bags in the subtree rooted at t. Let $K' = K \cap V_t$. Suppose $X \subseteq X_t$ and $\mathcal{P}$ is a partition of X. By definition, $c[t, X, \mathcal{P}]$ is the size

of a subgraph of type $(X_t, X, \mathcal{P}, K')$. Then, Lemmas 3.7 and 3.8 imply that $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, K')] = c[t, X, \mathcal{P}]$. In particular, for the root r of the tree decomposition, $\mathcal{A}[\ell_r, (\{z^*\}, \{z^*\}, \{\{z^*\}\}, K)] = c[r, \{z^*\}, \{\{z^*\}\}]$. Notice that $c[r, \{z^*\}, \{\{z^*\}\}]$ is the size of a minimum Steiner tree of G.

However, by Lemma 3.7, for all $i \in [N]$, if $\mathcal{A}[i, (\{z^*\}, \{z^*\}, \{\{z^*\}\}, K)] = \ell$, then there is a subgraph F that connects all terminals of K and that satisfies $\text{recdist}(F) \leq \ell$. As the algorithm outputs $\min_{i \in [N]} \mathcal{A}[i, (\{z^*\}, \{z^*\}, \{\{z^*\}\}, K)]$, it must output the weight of a minimum rectilinear Steiner tree of G. This proves the correctness of the algorithm.

The size of the table $\mathcal{A}[,]$ is $N \cdot 2^{O(\sqrt{n} \log n)}$, and each entry can be filled in time $2^{O(\sqrt{n} \log n)} n^{O(1)}$. Thus, the running time of the algorithm is $2^{O(\sqrt{n} \log n)} n^{O(1)}$. Using standard back-tracking tricks, we can also output an optimal RST. This concludes the proof. □

4 SUBEXPONENTIAL ALGORITHM FOR RECTILINEAR STEINER ARBORESCENCE

In this section, we give an outline for the subexponential algorithm for Rectilinear Steiner Arborescence, since the algorithm is essentially the same as that for Rectilinear Steiner Tree. We are again given K and the root vertex $r \in K$ as an input of Rectilinear Steiner Arborescence. Furthermore, let $|K| = n$ and G be the Hanan grid of K. We assume that the root vertex r is placed at $(0, 0)$ in $\mathbb{R}^2$. Recall that recdist_G is the weight function defined on the edges of G, and when the context becomes clear we simply denote it as recdist. The steps of the algorithm are very similar to those in the algorithm for Rectilinear Steiner Tree. Analogous to a shortest path RST, we define a rectilinear Steiner arborescence, called *shortest path RSA*, for a graph G. Then, for a shortest path RSA $\widehat{T}$, we show that there exists an optimal rectilinear Steiner arborescence T_{opt} such that the treewidth of $\widehat{T} \cup T_{\text{opt}}$ is $O(\sqrt{n})$. We then use this information to design our dynamic programming algorithm for Rectilinear Steiner Arborescence.

4.1 Shortest Path RSA and Its Properties

For a given G, we define a *shortest path RSA* similar to the definition of a shortest path RST. The definition is given next for completeness.

We give an arbitrary ordering $\{r, z_1, \ldots z_k\}$ on the terminals such that the root is the first vertex in the ordering. We define a shortest path RSA, $\widehat{S}$, through a constructive greedy process. Initially, we set T_1 to a $r - z_1$ monotone path. This is a rectilinear Steiner arborescence of $\{r, z_1\}$. In the i-th step, we compute a rectilinear Steiner arborescence T_{i+1} of $\{z_1, \ldots z_{i+1}\}$ from T_i as follows. If $z_{i+1} \in V(T_i)$, then we set $T_{i+1} = T_i$. Otherwise, for each vertex $u \in T_i$ let ℓ_u^1 be the length of a shortest $u - z_{i+1}$ path. Let ℓ_u^2 be the length of the shortest $r - u$ path that exists in T_i. In addition, ℓ denotes the length of a shortest $r - z_{i+1}$ path. Let $N \subseteq V(T_i)$ be the set of vertices such that for each $u \in N$, $\ell_u^1 + \ell_u^2 = \ell$. Let $u^* \in N$ be a vertex for which $\ell_{u^*}^1$ is minimized. If there is only one monotone $z_{i+1} - u^*$ path, then let Q be that path. Otherwise, there are two monotone $z_{i+1} - u^*$ paths such that one path has a horizontal edge incident with u^* and other has a vertical edge incident with u^*. If there is a horizontal edge in T_i that is incident with u^*, then we choose Q to be the monotone $z_{i+1} - u^*$ path such that the edge in Q incident with u^* is a horizontal edge. Otherwise, we choose Q to be the monotone $z_{i+1} - u^*$ path such that the edge in Q incident with u^* is a vertical edge. Then, we construct T_{i+1} by adding the monotone path Q to T_i. After $n - 1$ iterations, we construct a tree $\widehat{T} = T_n$ of G, which is a Steiner arborescence of K. This is our shortest path RSA.

By arguments similar to Lemma 3.1, it is possible to construct a shortest path RSA in polynomial time.

Lemma 4.1. *Given a set K of terminal points, and the Hanan grid G of K, a shortest path RSA $\widehat{T}$ of K can be constructed in polynomial time.*

Similar to Lemma 3.2, we can give a bound on the bend vertices of a shortest path RSA.

Lemma 4.2. *The number of bend vertices in $\widehat{T}$ is at most n.*

4.2 Supergraph of an Optimal RSA with Bounded Treewidth

Let K be an input set of n points for Rectilinear Steiner Arborescence, $r \in K$ is a root terminal, and G is the Hanan grid of K. In this part, given a shortest path RSA $\widehat{T}$, we show the existence of an optimum rectilinear Steiner arborescence T_{opt} of K with the property that the treewidth of $\widehat{T} \cup T_{\text{opt}}$ is $O(\sqrt{n})$. Similar to Lemma 3.3, we can show that there is an optimal rectilinear Steiner arborescence with a bounded number of bend vertices.

Lemma 4.3. *Let K be a set of n terminals in $\mathbb{R}^2$, with a root terminal $r \in K$ and G the Hanan grid of K. Then there is an optimum rectilinear Steiner arborescence of K in G such that the number of bend vertices of the Steiner arborescence in G is at most $3n$.*

After finding a shortest path RSA as described by Lemma 4.1, we prove the following structural lemma. This lemma is analogous to Lemma 3.4 but with some points of deviation due to the problem definition. For completeness, we give the additional details of the proof in the following.

Lemma 4.4. *Given a set K of n points, with $r \in K$ as the root terminal, and a shortest path RSA $\widehat{T}$ of K, there is an optimal rectilinear Steiner arborescence T_{opt} of K with the property that the treewidth of $\widehat{T} \cup T_{\text{opt}}$ is at most $41\sqrt{n}$.*

Proof. We choose an optimum rectilinear Steiner arborescence of K and prove it satisfies the required property. Among the optimum Steiner arborescences with minimum number of bend vertices, we select an arborescence T_{opt} that has maximum edge intersection with $\widehat{T}$.

We show that $\widehat{T} \cup T_{\text{opt}}$ has $O(\sqrt{n})$ treewidth. For the sake of contradiction, suppose $\widehat{T} \cup T_{\text{opt}}$ has treewidth greater than $41\sqrt{n}$. Again, we can assume that $n \geq 4$, as otherwise we can greedily find out the best rectilinear Steiner arborescence from the constant-sized Hanan grid. Then, by Proposition 2.7, there is a $9\sqrt{n} \times 9\sqrt{n}$ grid H appearing as a minor in $\widehat{T} \cup T_{\text{opt}}$. Let $\mathcal{P}(H) = \{C_v | v \in V(H)\}$ be a minor model of H. For a vertex $v \in V(H)$, if any vertex of C_v is a terminal vertex of G, a bend vertex of $\widehat{T}$, or a bend vertex of T_{opt}, then we mark the vertex v in H. By Lemmas 4.2 and 4.3, the number of vertices of H that get marked is at most $5n$. By the arguments given in Lemma 3.4, we can find from H a 2×2 grid H' in H, where none of the vertices are marked. With arguments similar to Claim 2, we can also find a subgrid G' in G (in some sense contained in H') that has the following properties:

(1) No vertex in G' is a terminal vertex of G or a bend vertex for $\widehat{T}$ or T_{opt}.
(2) There are four vertices $u_1, \ldots, u_4$ that are of degree exactly 4 in $\widehat{T} \cup T_{\text{opt}}$. All other vertices are of degree exactly 2 in $\widehat{T} \cup T_{\text{opt}}$.
(3) There are horizontal paths $P_{12} = u_1 - u_2$, $P_{34} = u_3 - u_4$ and vertical paths $P_{13} = u_1 - u_3$, $P_{24} = u_2 - u_4$. Each of the internal vertices of these paths are of degree 2 in the grid G.
(4) Either all horizontal paths belong to T_{opt} and not $\widehat{T}$ or all vertical paths belong to exactly $\widehat{T}$ and not T_{opt}, or vice versa. These are the only two possibilities.

The following observation tells us about the position of the vertices in G', with respect to the origin, where the root terminal is positioned. □

Observation 7. *Let K be a set of terminals with the root terminal r placed at the origin. Let G be the Hanan grid of K and T_{mml} be an edge minimal Steiner arborescence for K. Let $u, v \in V(T_{\text{mml}})$ be a pair of vertices with the following properties:*

- *Either* $u_x = v_x \neq r_x$ *and* $u_y < r_y < v_y$ *or* $u_y = v_y \neq r_y$ *and* $u_x < r_x < v_x$.
- *The monotone* $u - v$ *path* Q *is a subgraph of* T_{mml}.

Then it cannot be the case that all internal vertices of Q *have degree 2 in* T_{mml}.

Proof. Without loss of generality, assume that $u_x = y_x \neq r_x$ and $u_y < r_y < v_y$ are true. For the sake of contradiction, let the monotone $u - v$ path Q be a subgraph of T_{mml} such that every internal vertex of Q is degree 2 in T_{mml}. Let T' be the graph obtained by deleting the internal vertices and edges of Q. Since for each $z \in K$ there is a unique $r - z$ path in T_{mml}, if z is still connected to r in T', then the $r - z$ path is still a shortest $r - z$ path. We show that all terminals are still connected to r in T'. This implies that T' is a Steiner arborescence, thereby contradicting the edge minimality of T_{mml}.

Suppose there is a terminal z such that the $r - z$ path Q' of T_{mml} had an edge in common with $E(Q)$. Since every internal vertex of Q is of degree 2 in T_{mml}, it must be the case that the entire path Q is a subpath of Q'. By definition of Q, the entire path cannot be contained in the grid defined by r and z. However, from Observation 5, it cannot be the case that Q is a shortest $r - z$ path. Thus, for no terminal z, does the $r - z$ path in T_{mml} intersect with Q. Thus, each terminal z remains connected to r in T'. Hence, we conclude that such a path Q cannot exist in an edge minimal Steiner arborescence T_{mml}. □

By definition, both $\widehat{T}$ and T_{opt} are minimal rectilinear Steiner arborescences. Then, by Observation 7, it follows that G' lies in one of the quadrants of $\mathbb{R}^2$. For the sake of the proof, we assume without loss of generality that G' lies in the first quadrant of $\mathbb{R}^2$. In addition, without loss of generality, let the horizontal paths belong to T_{opt} and the vertical paths belong to $\widehat{T}$. Let the length of P_{12} be ℓ. By the definition of the subgrid G', the length of P_{34} is also ℓ. Let the length of P_{13} be p. This is also the length of P_{24}. Suppose $\ell > p$. Then, we consider the Steiner arborescence formed by deleting, in T_{opt}, the path P_{12} and adding the path P_{24}. The resulting Steiner arborescence has weight strictly less than that of weight of T_{opt}. This contradicts the choice of T_{opt}. Hence, this is not possible.

Now, suppose $\ell \leq p$. Consider the two paths P_{13} and P_{24}. They are paths of $\widehat{T}$. By construction and by Observation 4, they cannot be subpaths of a path added in a single construction step, as otherwise they will not be part of a shortest path. Also by construction, one of them is added to $\widehat{T}$ before the other. Without loss of generality, let P_{13} be added before P_{24}. By construction, P_{24} is a subpath of a path P that was added in some step i, to connect a terminal z to the already constructed T_{i-1}. By definition of H' and G', this terminal must lie outside the region formed by the subgrid G'. Since P_{24} was part of a shortest path between T_{i-1} and z, z must lie on a row strictly higher than the rows of G'. Since, u_1 and u_2 are not bend vertices in $\widehat{T}$, they have neighbors u_1' and u_2' in $\widehat{T}$. In addition, u_1' and u_2' are on the same row, and u_1' is on the same column as u_1, whereas u_2' is on the same column as u_2. Let $P_{u_1'}$ be the path from r to u_1' in T_{i-1}, $P_{u_2'}$ be the subpath of P between r and u_2', and P_z be the subpath of P between u_2' and z. By definition of a shortest path, the subpath $P_{u_2'}$ is a shortest path between r and u_2', and the subpath P_z is a shortest path between u_2' and z. Let $G_{u_1'} \leq_s G$ be the grid defined by r, u_1' as its diagonal points and $G_{u_2'} \leq_s G$ be the grid defined by r, u_2' as its diagonal points. Due to the position of the vertex r, $G_{u_1'} \leq_s G_{u_2'}$. Then, by Observation 5, $P_{u_1'} \cup P'$ is a shortest path between r and u_2', as is $P_{u_2'}$. This implies that $P_{u_1'} \cup P' \cup P_z$ is a shortest $r - z$ path. Notice that P is of weight at least (u_2', u_2, u_4). Since $\ell \leq p$, this implies that P is of weight strictly more that the path P'. However, by the description of construction of $\widehat{T}$, the path $P' \cup P_z$ is a better candidate than the path P in step i. This contradicts the choice of adding the path P to form T_i. Therefore, it is not possible that $\ell \leq p$.

Thus, we obtain a contradiction to the fact that $\widehat{T} \cup T_{\text{opt}}$ has a grid minor greater than $9\sqrt{n}$. This proves that $\widehat{T} \cup T_{\text{opt}}$ has treewidth at most $41\sqrt{n}$.

4.3 Dynamic Programming Algorithm for Rectilinear Steiner Arborescence: An Overview

In this section, we describe the general ideas of the subexponential algorithm for Rectilinear Steiner Arborescence, since it is very similar to the algorithm for Rectilinear Steiner Tree. The full details of the algorithm are given in Appendix A. Due to Lemma 4.4, we know that given a shortest path RSA $\widehat{T}$, there exists an optimal Steiner arborescence T_{opt} such that the treewidth of $T = \widehat{T} \cup T_{\text{opt}}$ is at most $41\sqrt{n}$. Note that we do not know the subgraph T. However, we simulate a dynamic programming algorithm by storing all of the information needed to compute an optimal Steiner arborescence on the tree decomposition of T.

Again, if we knew the subgraph T of G such that there is an optimal Steiner arborescence fully contained in T, we could design a dynamic programming algorithm for Rectilinear Steiner Arborescence over the tree decomposition of T. This algorithm is similar in ideas to the algorithm for Steiner Tree over a tree decomposition of a given graph. However, again the difficulty for us is that we do not know T. Suppose we had $(\mathbb{T}, \mathcal{X}' = \{X'_t\}_{t \in V(\mathbb{T})})$, which is a nice tree decomposition of T, of width at most $41\sqrt{n}$, where $\mathbb{T}$ is a rooted tree. Let the root node be $\tilde{t}$. From this, we obtain a new tree decomposition $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ by adding the input root terminal r to each bag X'_t, $t \in V(\mathbb{T})$. We continue to name a bag X_t as we named X'_t—for instance, if X'_t was a leaf bag, then so is X_t and so on. Notice that the treewidth of $(\mathbb{T}, \mathcal{X})$ increases by at most 1, but the root and leaf bags of $\mathbb{T}$ are identical to the singleton set $\{r\}$. For a node t, let V_t be the union of all bags present in the subtree of $\mathbb{T}$ rooted at t. For a node t, we define a graph $G_t = (V_t, E_t = \{e \in G : e \text{ is introduced in the subtree rooted at } t\})$. A relevant definition for this problem is that of a locally rooted subgraph. A forest $F \leq_s G$, with connected components $C_1, \ldots, C_q$, is called a *locally rooted subgraph* if each component C_i has a special vertex or a root vertex r_i. Let us give a brief intuition behind the definition of a locally rooted subgraph. Consider an optimal rectilinear Steiner arborescence of T, and let F be the restriction in the subgraph G_t. Then F will be a forest $C_1, \ldots, C_q$. Moreover, by definition of an arborescence, in each component C_i there is a unique vertex r_i through which the unique shortest path between r and any vertex $v \in C_i$ passes. Thus, F can be thought of as a locally rooted subgraph. The aim is to again build an optimal rectilinear Steiner arborescence bottom-up from the leaf bags of the tree decomposition.

Thus, the important step in the algorithm for Rectilinear Steiner Arborescence is to compute the following information about locally rooted subgraphs: for each bag X_t, $X \subseteq X_t$, a partition $\mathcal{P} = (P_1, \ldots, P_q)$ of X, and a set $X_{\text{sp}} = \{r_1, \ldots, r_q\}$ such that $r_i \in P_i$, for each $i \in [q]$, the value $c[t, X, \mathcal{P}, X_{\text{sp}}]$ is the minimum weight of a locally rooted subgraph F of G_t with the following properties:

(1) F has exactly q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_i = X_t \cap V(C_i)$ for all $i \in [q]$. In other words, $\mathcal{P}$ corresponds to connected components of F.
(2) $X_t \cap V(F) = X$. In other words, the vertices of $X_t \setminus X$ are untouched by F.
(3) $K \cap V_t \subseteq V(F)$. In other words, all terminal vertices in G_t belong to F.
(4) For each $i \in [q]$, and each vertex $w \in V(C_i) \setminus \{r_i\}$, the $w - r_i$ path in F is a shortest path in G and there is a $r_i - r$ shortest path in G that has r_i as an internal vertex.

Suppose we know the values $c[t, X, \mathcal{P}, X_{\text{sp}}]$ for each tuple $(t, X, \mathcal{P}, X_{\text{sp}})$, where $t \in V(\mathbb{T}), X \subseteq X_t$, $\mathcal{P}$ is a partition of X, and X_{sp} is a set of vertices such that there is exactly one vertex from each block of $\mathcal{P}$. Then by definition, $c[\tilde{t}, \{r\}, \{\{r\}\}, \{r\}]$ corresponds to the weight of a minimum

Steiner arborescence. Thus, knowing the function $c[\cdot]$ is enough to know the weight of an optimal rectilinear Steiner arborescence of K in T. In our case, we do not know the graph $T = \widehat{T} \cup T_{\text{opt}}$ and a tree decomposition of T, and therefore we do not know how to calculate the function $c[\cdot]$. However, we know that the treewidth of T is at most $41\sqrt{n}$. As argued in the case of Rectilinear Steiner Tree, this implies that the number of choices for bags in a tree decomposition of T is at most $n^{O(\sqrt{n})}$. Notice that properties (1) through (3) are the same as the properties maintained for calculating the function $c[\cdot]$ for Rectilinear Steiner Tree on a tree decomposition of T. Property (4) is necessary to solve the Rectilinear Steiner Arborescence problem. Each vertex $r_i \in X_{\text{sp}}$ can be thought of as a local root vertex for the component C_i of F.

Now that we do not know the graph T and therefore do not have a desired tree decomposition of T, we work around this bottleneck as in the case of Rectilinear Steiner Tree by defining *types*. A *type* is the analogue of a tuple $(t, X, \mathcal{P}, X_{\text{sp}})$, as input for the function $c[\cdot]$ defined earlier for Rectilinear Steiner Arborescence, such that the following hold:

(i) Y is a subset of $V(G)$ of size at most $41\sqrt{n} + 2$.
(ii) $\mathcal{P}$ is a partition of Y' with q blocks.
(iii) There exists a set of components $C_1, \ldots, C_q$ in $\widehat{T} \setminus Y$ such that $K' = K \cap (Y' \cup \bigcup_{i=1}^{q} V(C_i))$.
(iv) Y_{sp} has exactly one vertex r_i from each block $P_i \in \mathcal{P}$.

In a type $(Y, Y', \mathcal{P}, Y_{\text{sp}}, K')$, Y represents a potential bag Y of a node (say t) in a tree decomposition of T. The set Y', partition $\mathcal{P}$, and Y_{sp} have the same meaning as that in the tuple $(t, Y', \mathcal{P}, Y_{\text{sp}})$ for the function $c[\cdot]$ for Rectilinear Steiner Arborescence over a tree decomposition of an input graph. The set K' is the set of terminals in the graph G_t. Again, similar to Lemma 3.6, we can show that the number of types for Rectilinear Steiner Arborescence is bounded by $2^{O(\sqrt{n}\log n)} n^{O(1)}$ and that the set of all types can be found in as much time.

In the following, we explain the steps of our algorithm, which are very similar to that of the Rectilinear Steiner Tree algorithm. The differences in the two algorithms are mainly technicalities due to the differences in the definitions of the two problems. Otherwise, the ideas for both algorithms are very similar. We fix an integer $N = |3(V(G)| + |E(G)|)$. Just like the algorithm for Rectilinear Steiner Tree, this algorithm computes values $\mathcal{A}[i, D]$, where $i \in [N]$ and D is a type. As before, we want the table $\mathcal{A}[,]$ to contain all of the information that is necessary for computing the function $c[\cdot]$ defined earlier for Rectilinear Steiner Tree, over a tree decomposition of T. Since we do not know the graph T, we assume a hypothetical nice tree decomposition $\mathcal{T} = (\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ of T of width at most $41\sqrt{n}$. We may also assume that the nice tree decomposition is rooted at a node $\tilde{t} \in \mathbb{T}$. Recall that the level of a vertex $t \in \mathbb{T}$ is the height of the subtree of $\mathbb{T}$ rooted at t. From Proposition 2.1, the level of any node of $\mathbb{T}$ is at most N. Suppose $t \in \mathbb{T}$ is a node at level i and corresponds to the bag X_t. Let V_t be the union of bags present in the subtree rooted at t. Let the graph G_t be defined as $(V_t, \{e \mid e \text{ is introduced in the subtree rooted at } t\})$. Let $K' = V_t \cap K$. Then, for any $X \subseteq X_t$, a partition $\mathcal{P}$ of X, and a set X_{sp} defined with exactly one vertex from each block of $\mathcal{P}$, $\mathcal{A}[i, (X_t, X, \mathcal{P}, X_{\text{sp}}, T')] = c[t, X, \mathcal{P}, X_{\text{sp}}]$. As mentioned before, $c[t, X, \mathcal{P}, X_{\text{sp}}]$ is the minimum weight of the locally rooted subgraph F of G_t such that the following hold: (i) F has q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_j = X_t \cap V(C_j)$, (ii) $X_t \cap V(F) = X$, (iii) $K \cap V_t \subseteq V(F)$, and (iv) for each $j \in [q]$, and each vertex $w \in V(C_j) \setminus \{r_j\}$, and each vertex $w \in V(C_j) \setminus \{r_j\}$, the $w - r_j$ path in F is a shortest path in G and there is a $r_j - r$ shortest path in G that has r_j as an internal vertex. For other pairs (i, D), we do not guarantee that the value of $\mathcal{A}[i, D]$ is meaningful. As we had motivated the ideas behind the algorithm for Rectilinear Steiner Tree, the main idea behind this algorithm is that we pretend that a tree

decomposition for T is given to us, even though in reality we do not even know the graph T. The full details of the algorithm can be found in Appendix A.

In the end, we obtain our desired subexponential algorithm.

Theorem 4.5. Rectilinear Steiner Arborescence *can be solved in time* $2^{O(\sqrt{n}\log n)}n^{O(1)}$.

5 CONCLUSION

We exhibit deterministic subexponential algorithms for Rectilinear Steiner Tree and Rectilinear Steiner Arborescence. Both algorithms run in $2^{O(\sqrt{n}\log n)}n^{O(1)}$ time. Finding a lower bound for the running time of an algorithm and exhibiting an algorithm with optimal running time, for both problems, remain open for investigation.

APPENDIX

A DYNAMIC PROGRAMMING ALGORITHM FOR RECTILINEAR STEINER ARBORESCENCE

For the sake of completeness, in this section we provide full details of the subexponential algorithm for Rectilinear Steiner Arborescence. Recall from Lemma 4.4 that given a shortest path RSA $\widehat{T}$, there exists an optimal Steiner arborescence T_{opt} such that the treewidth of $T = \widehat{T} \cup T_{\text{opt}}$ is at most $41\sqrt{n}$. As in the case of the Rectilinear Steiner Tree problem, we do not know the graph T. However, our objective is to design a dynamic programming algorithm where each state contains all of the information needed to compute an optimal Steiner arborescence on the tree decomposition of T.

Let $(\mathbb{T}, \mathcal{X}' = \{X'_t\}_{t\in V(\mathbb{T})})$ be a potential nice tree decomposition of T, of width at most $41\sqrt{n}$, where $\mathbb{T}$ is a rooted tree. Let the root node be $\tilde{t}$. From this, we obtain a new tree decomposition $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t\in V(\mathbb{T})})$ by adding the input root terminal r to each bag X'_t, $t \in V(\mathbb{T})$. We continue to name a bag X_t as we named X'_t—that is, if X'_t was a leaf bag, then so is X_t and so on. Notice that the treewidth of $(\mathbb{T}, \mathcal{X})$ increases by at most 1, but the root and leaf bags of $\mathbb{T}$ are identical to the singleton set $\{r\}$. For a node t, let V_t be the union of all bags present in the subtree of $\mathbb{T}$ rooted at t. For a node t, we define a graph $G_t = (V_t, E_t = \{e \in G : e \text{ is introduced in the subtree rooted at } t\})$. Recall the definition of a *locally rooted subgraph*. A forest $F \leq_s G$, with connected components $C_1, \ldots, C_q$, is called a *locally rooted subgraph* if each component C_i has a special vertex or a root vertex r_i. The reason for considering locally rooted subgraphs was as follows. Consider an optimal rectilinear Steiner arborescence of T, and let F be the restriction in the subgraph G_t. Then, F will be a forest $C_1, \ldots, C_q$. Moreover, by definition of an arborescence, in each component C_i there is a unique vertex r_i through which the unique shortest path between r and any vertex $v \in C_i$ passes. Thus, F can be thought of as a locally rooted subgraph. We wish to build an optimal rectilinear Steiner arborescence bottom-up from the leaf bags of the tree decomposition.

To do this, we first look at a function $c[\cdot]$ for Rectilinear Steiner Arborescence: for each bag X_t, $X \subseteq X_t$, a partition $\mathcal{P} = (P_1, \ldots, P_q)$ of X, and a set $X_{\text{sp}} = \{r_1, \ldots, r_q\}$ such that $r_i \in P_i$, for each $i \in [q]$, the value $c[t, X, \mathcal{P}, X_{\text{sp}}]$ is the minimum weight of a locally rooted subgraph F of G_t with the following properties:

(1) F has exactly q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_i = X_t \cap V(C_i)$ for all $i \in [q]$. In other words, $\mathcal{P}$ corresponds to connected components of F.
(2) $X_t \cap V(F) = X$. In other words, the vertices of $X_t \setminus X$ are untouched by F.
(3) $K \cap V_t \subseteq V(F)$. In other words, all terminal vertices in G_t belong to F.
(4) For each $i \in [q]$, and each vertex $w \in V(C_i) \setminus \{r_i\}$, the $w - r_i$ path in F is a shortest path in G and there is a $r_i - r$ shortest path in G that has r_i as an internal vertex.

Again, the number of blocks in $\mathcal{P}$ is q, and this variable is used throughout the section to denote the number of blocks of the partition in question. Suppose we know the values $c[t, X, \mathcal{P}, X_{\text{sp}}]$ for each tuple $(t, X, \mathcal{P}, X_{\text{sp}})$, where $t \in V(\mathbb{T}), X \subseteq X_t$, $\mathcal{P}$ is a partition of X, and X_{sp} is a set of vertices such that there is exactly one vertex from each block of $\mathcal{P}$. Then by definition, $c[\tilde{t}, \{r\}, \{\{r\}\}, \{r\}]$ corresponds to the weight of a minimum Steiner arborescence. Thus, knowing the function $c[\cdot]$ is enough to know the weight of an optimal rectilinear Steiner arborescence of K in T. In our case, we do not know the graph $T = \widehat{T} \cup T_{\text{opt}}$ and a tree decomposition of T, but we know that the treewidth of T is at most $41\sqrt{n}$. As argued in the case of Rectilinear Steiner Tree, this implies that the number of choices for bags in a tree decomposition of T is at most $n^{O(\sqrt{n})}$. Notice that properties (1) through (3) are same as the properties for defining the function $c[\cdot]$ for Rectilinear Steiner Tree on a tree decomposition of T. Property (4) is necessary to solve the Rectilinear Steiner Arborescence problem. Each vertex $r_i \in X_{\text{sp}}$ can be thought of as a local root vertex for the component C_i of F. Let us repeat the intuition behind this property. Suppose F was a subgraph of an edge minimal Steiner arborescence T^* rooted at r. In addition, for each $i \in [q]$, r_i is the unique vertex in C_i that has minimum distance to r in T. Then, for each $i \in [q], w \in C_i$, the $w - r$ path in T^* satisfies property (4). In particular, if T^* was an optimal Steiner arborescence, property (4) holds for F. The number of choices for X_{sp}, which is a subset of X, is at most $2^{\sqrt{n}}$.

In fact, if there are two vertices u, v such that there is a $u - r$ shortest path in G that has v as an internal vertex, we say that u has the *shortness property* with v. Notice that the shortness property is transitive. In other words, if u has the shortness property with v and w has the shortness property with u, then w has the shortness property with v. Whether a vertex u has the shortness property with v can be tested by checking if, in G, the length of a shortest $u - v$ path plus the length of a shortest $v - r$ path equals the length of a shortest $u - r$ path.

We explain the algorithm more formally. We first define a *type* that is the analogue of a tuple $(t, X, \mathcal{P}, X_{\text{sp}})$, as input for the function $c[\cdot]$ defined previously for Rectilinear Steiner Arborescence.

Definition A.1. A type is a tuple $(Y, Y' \subseteq Y, \mathcal{P}, Y_{\text{sp}}, K')$ such that the following hold:

(i) Y is a subset of $V(G)$ of size at most $41\sqrt{n} + 2$.
(ii) $\mathcal{P}$ is a partition of Y' with q blocks.
(iii) There exists a set of components $C_1, \ldots, C_q$ in $\widehat{T} \setminus Y$ such that $K' = K \cap (Y' \cup \bigcup_{i=1}^{q} V(C_i))$.
(iv) Y_{sp} has exactly one vertex r_i from each block $P_i \in \mathcal{P}$.

In a type $(Y, Y', \mathcal{P}, Y_{\text{sp}}, K')$, Y represents a potential bag Y of a node (say t) in a tree decomposition of T. The set Y', partition $\mathcal{P}$, and Y_{sp} have the same meaning as that of the tuple $(t, Y', \mathcal{P}, Y_{\text{sp}})$ for the function $c[\cdot]$ for Rectilinear Steiner Arborescence over a tree decomposition of an input graph. The set K' is the set of terminals in the graph G_t. We can show that the number of types is bounded.

Lemma A.2. *There is a $2^{O(\sqrt{n}\log n)} n^{O(1)}$ time algorithm $\mathcal{B}$ enumerating all of the types.*

Proof. The number of choices for Y is $n^{\sqrt{n}}$. Once a Y is fixed, the number of choices for Y' is $O(2^{\sqrt{n}})$, whereas the number of choices for the partition $\mathcal{P}$, of Y', is $O(\sqrt{n}^{\sqrt{n}})$. By definition of the Hanan grid G, each vertex in G has at most four neighbors. Thus, $\widehat{T} - Y$ has at most $4\sqrt{n}$ components. On fixing a Y, the choices for K' correspond to the $2^{O(\sqrt{n})}$ choices of at most $4\sqrt{n}$ components of $\widehat{T} - Y$. This gives us the desired bound on the number of suitable types. □

In the following, we explain the steps of our algorithm, which are very similar to that of the Rectilinear Steiner Tree algorithm. The differences in the two algorithms are mainly technicalities due to the differences in the definitions of the two problems. Otherwise, the ideas for both the algorithms are very similar. For the sake of completeness, we give the full algorithm for Rectilinear Steiner Arborescence in this article. We fix an integer $N = |3(V(G)| + |E(G)|)$. Just like the dynamic programming algorithm for Rectilinear Steiner Tree, this algorithm computes values $\mathcal{A}[i, D]$, where $i \in [N]$ and D is a type. As before, we want the table $\mathcal{A}[,]$ to contain all of the information that is necessary for correctly computing the function $c[\cdot]$ for Rectilinear Steiner Arborescence. Since we do not know the graph T, we assume a hypothetical nice tree decomposition $\mathcal{T} = (\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ of T of width at most $41\sqrt{n}$. We may also assume that the nice tree decomposition is rooted at a node $\tilde{t} \in \mathbb{T}$. Recall that the level of a vertex $t \in \mathbb{T}$ is the height of the subtree of $\mathbb{T}$ rooted at t. From Proposition 2.1, the level of any node of $\mathbb{T}$ is at most N. Suppose $t \in \mathbb{T}$ is a node at level i and corresponds to the bag X_t. Let V_t be the union of bags present in the subtree rooted at t. Let the graph G_t be defined as $(V_t, \{e \mid e \text{ is introduced in the subtree rooted at } t\})$. Let $K' = V_t \cap K$. Then, for any $X \subseteq X_t$, a partition $\mathcal{P}$ of X, and a set X_{sp} defined with exactly one vertex from each block of $\mathcal{P}$, $\mathcal{A}[i, (X_t, X, \mathcal{P}, X_{\text{sp}}, T')] = c[t, X, \mathcal{P}, X_{\text{sp}}]$. As mentioned before, $c[t, X, \mathcal{P}, X_{\text{sp}}]$ is the minimum weight of the locally rooted subgraph F of G_t such that the following hold: (i) F has q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_j = X_t \cap V(C_j)$, (ii) $X_t \cap V(F) = X$, (iii) $K \cap V_t \subseteq V(F)$, and (iv) for each $j \in [q]$, and each vertex $w \in V(C_j) \setminus \{r_j\}$, the $w - r_j$ path in F is a shortest path in G and w has the shortness property with r_j. For other pairs (i, D), we do not guarantee that the value of $\mathcal{A}[i, D]$ is meaningful. As we had motivated the ideas behind the algorithm for Rectilinear Steiner Tree, the main idea behind this algorithm is that we pretend that a tree decomposition for T is given to us, even though in reality we do not even know the graph T.

We write a recurrence relation for $\mathcal{A}[i, D]$, where $i \in [N]$ and D is a type. The motivation for the recurrence relation is similar to that for the recurrence in the subexponential algorithm of Rectilinear Steiner Tree.

$$\mathcal{A}[1, D] = \begin{cases} 0 & \text{if } D = (\{r\}, \{r\}, \{\{r\}\}, \{r\}, \{r\}) \\ \infty & \text{otherwise} \end{cases} \tag{7}$$

To define $\mathcal{A}[i, D]$ for $i \in [N] \setminus \{1\}$ and a type $D = (Y, Y', \mathcal{P}, Y_{\text{sp}}, K')$, we first define many intermediate values and take the minimum over all such values.

For all $v \in Y$,

$$I_v[i, D] = \begin{cases} \infty & \text{if } v \notin Y' \text{ and } v \in K \\ \infty & \text{if } v \in Y' \text{ but } \{v\} \notin \mathcal{P} \\ \mathcal{A}[i-1, (Y \setminus \{v\}, Y', \mathcal{P}, Y_{\text{sp}}, K')] & \text{if } v \notin Y' \text{ and } v \notin K \\ \mathcal{A}[i-1, (Y \setminus \{v\}, Y' \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, Y_{\text{sp}} \setminus \{v\}, K' \setminus \{v\})] & \text{if } v \in Y'. \end{cases} \tag{8}$$

Intuitively, if Y is a bag corresponding to a node t in a tree decomposition of T and K' is the set of terminals in G_t, then Equation (8) corresponds to the computation of the function $c[t, Y', \mathcal{P}, Y_{\text{sp}}]$ of Rectilinear Steiner Arborescence.

For all $u, v \in Y$ such that u, v belong to the same block P of $\mathcal{P}$ and $uv \in E(G)$, we do the following. Let w be the vertex that belongs to $P \cap Y_{\text{sp}}$. We first define a set $\mathbb{P}$ of pairs $(\mathcal{P}', Y'_{\text{sp}})$ on Y'. For a pair $(\mathcal{P}', Y'_{\text{sp}})$, $\mathcal{P}'$ denotes a partition of Y', whereas Y'_{sp} is defined by taking exactly one vertex per block of $\mathcal{P}'$. For a pair $(\mathcal{P}', Y'_{\text{sp}})$ to belong to $\mathbb{P}$, they must be exactly one of the following kinds of pairs:

(1) The vertices u, v belong to the same block of $\mathcal{P}'$. For each block $P' \in \mathcal{P}'$, and vertex $v \in P' \cap Y'_{\text{sp}}$, every vertex of P' has the shortness property with v.

(2) The vertices u, v belong to distinct blocks P_u, P_v, respectively, of $\mathcal{P}'$. For each block $P' \in \mathcal{P}'$, and vertex $v \in P' \cap Y'_{\text{sp}}$, every vertex of P' has the shortness property with v. Assume $u^* \in Y'_{\text{sp}} \cap P_u$ and $v^* \in Y'_{\text{sp}} \cap P_v$. Then, exactly one of the two possibilities holds:
- It holds that $u^* = w$, $v^* = v$. The vertex v has the shortness property with u and therefore the shortness property with w.
- It holds that $v^* = w$, $u^* = v$. The vertex u has the shortness property with v and therefore the shortness property with w.

Then, for the pair $u, v \in Y$,

$$I_{uv}[i, D] = \min \left\{ \min_{(\mathcal{P}', Y'_{\text{sp}}) \in \mathbb{P}} \left\{ \mathcal{A}[i-1, (Y, Y', \mathcal{P}', Y'_{\text{sp}}, K')] + \text{recdist}(uv) \right\}, \mathcal{A}[i-1, D] \right\}. \quad (9)$$

Note that if $\{u, v\} \nsubseteq Y'$ or u and v are in same block of $\mathcal{P}$, then Equation (9) gives $I_{uv}[i, D] = \mathcal{A}[i-1, D]$. Equation (9) corresponds to the computation of values in the introduce edge node where the edge uv is introduced.

For all $w \in V(G)$,

$$F_w[i, D] = \min \left\{ \min_{\mathcal{P}'} \left\{ \mathcal{A}[i-1, (Y \cup \{w\}, Y' \cup \{w\}, \mathcal{P}', Y_{\text{sp}}, K')] \right\}, \mathcal{A}[i-1, (Y \cup \{w\}, Y', \mathcal{P}, Y_{\text{sp}}, K')] \right\}, \quad (10)$$

where $\mathcal{P}'$ in the inner minimum varies over all partitions obtained by adding w to one of the existing blocks, and w was not a local root vertex. Equation (10) corresponds to computation in a forget node where w is forgotten.

Let $\mathbb{Q}$ be the set of tuples, $(\mathcal{P}_1, \mathcal{P}_2, Y^1_{\text{sp}}, Y^2_{\text{sp}})$, that satisfies the following properties:

(1) $\mathcal{P}_1 \sqcup \mathcal{P}_2 = \mathcal{P}$.
(2) $Y_{\text{sp}} \subseteq Y^1_{\text{sp}} \cup Y^2_{\text{sp}}$.
(3) For $i \in [2]$, Y^i_{sp} is defined with exactly one vertex from each block of $\mathcal{P}_i$. All vertices in a block of $\mathcal{P}_i$ have the shortness property with the vertex of Y^i_{sp} in that block.
(4) Let P_1 and P_2 be blocks of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. Then $|P_1 \cap P_2| \leq 1$. A vertex in $P_1 \cap P_2$ belongs to at least one of Y^1_{sp} and Y^2_{sp}. We call such a vertex an intersection vertex.
(5) For any block $P \in \mathcal{P}$, let it be formed by $\{P_{11}, P_{12}, \ldots, P_{1a}\} \in \mathcal{P}_1$ and $\{P_{21}, P_{22}, \ldots, P_{2b}\} \in \mathcal{P}_2$. Let Y'_1 be the subset of Y^1_{sp} defined by $\{P_{11}, P_{12}, \ldots, P_{1a}\}$. Let Y'_2 be the subset of Y^2_{sp} defined by $\{P_{21}, P_{22}, \ldots, P_{2b}\}$. Let $r_P = P \cap Y_{\text{sp}}$. Consider the auxiliary bipartite graph $H = (A \uplus B, E(H))$ where the vertices in A correspond to $\{P_{11}, P_{12}, \ldots, P_{1a}, P_{21}, P_{22}, \ldots, P_{2b}\}$ and $B = Y'_1 \cup Y'_2$. An edge is added between a vertex $u \in A$ and $v \in B$ if the block corresponding to u contains the intersection vertex corresponding to v. This auxiliary graph H must be a tree for $(\mathcal{P}_1, \mathcal{P}_2, Y^1_{\text{sp}}, Y^2_{\text{sp}})$ to be a tuple of $\mathbb{Q}$. For a vertex $v \in B$, let R_v be the $r_P - v$ path in H. Let $L_v = \{v = v_1, v_2, \ldots, r_P = v_\ell\}$ be the sequence of intersection vertices obtained from R_v. Then for two consecutive vertices $\{v_j, v_{j+1}\}$ in L_v, v_j has the shortness property with v_{j+1}.

With respect to the set $\mathbb{Q}$, we define the following equation.

$$J[i, D] = \min_{\substack{\mathcal{P} = \mathcal{P}_1 \sqcup \mathcal{P}_2 \\ (\mathcal{P}_1, \mathcal{P}_2, Y^1_{\text{sp}}, Y^2_{\text{sp}}) \in \mathbb{Q} \\ K'_1 \cup K'_2 = K' \\ i' \leq i-1}} \left\{ \mathcal{A}[i-1, (Y, Y', \mathcal{P}_1, Y^1_{\text{sp}}, K'_1)] + \mathcal{A}[i', (Y, Y', \mathcal{P}_2, Y^2_{\text{sp}}, K'_2)] \right\} \quad (11)$$

Equation (11) corresponds to a computation in a join node.

We define $\mathcal{A}[i, D]$ for $i \in [N] \setminus \{1\}$ and type $D = (Y, Y', \mathcal{P}, Y_{\text{sp}}, K')$ as follows.

$$\mathcal{A}[i, D] = \min \begin{cases} \min\limits_{v \in Y} I_v[i, D] \\ \min\limits_{\substack{uv \in E(G) \\ u, v \in Y}} I_{uv}[i, D] \\ \min\limits_{w \in V(G)} F_w[i, D] \\ J[i, D] \end{cases} \tag{12}$$

For each $i \in [N]$ and each type D, we associate with $A[i, D]$ a locally rooted subgraph of T. We say that a locally rooted subgraph F is of type $(Y, Y', \mathcal{P}, Y_{\text{sp}}, K')$, where $\mathcal{P} = \{P_1, \dots P_q\}$ if the following hold:

(a) The number of connected components in F is equal to $|\mathcal{P}| = q$. In addition, $V(C_j) \cap Y = P_j$.
(b) $V(F) \cap K = K'$.
(c) For each $j \in [q]$, $r_j \in P_j$.
(d) For each $j \in [q]$, and each $w \in C_j$, the $r_j - w$ path in F is a shortest path in G and there is a shortest $r - w$ path in G with r_j appearing as an internal vertex.

The next lemma shows the relation between the function $\mathcal{A}[\cdot]$ and the set of locally rooted subgraphs of T.

Lemma A.3. *Let $i \in [N]$ and D be a type. Furthermore, let $A[i, D]$ be computed by Equation (12), and have a finite value ℓ. Then there is a locally rooted subgraph F, of type D, such that* $\text{recdist}(F) \leq \ell$.

Proof. We show the statement using induction on i.

Case 1: Base case. Since the graph $(\{r\}, \emptyset)$ is of type $(\{r\}, \{r\}, \{\{r\}\}, \{r\}, \{r\})$, the base case holds trivially.

Now, let $1 < i \leq N$ and $D = (Y, Y', \mathcal{P}, Y_{\text{sp}}, K')$ be a type and $\mathcal{A}[i, D] = \ell$. We need to show that there is a locally rooted subgraph F of G such that F has type D and $\text{recdist}(F) \leq \ell$.

Case 2. We know that $\mathcal{A}[i, D]$ is computed using Equation (12), which is a minimum over a set of values. Suppose $\mathcal{A}[i, D] = I_v[i, D] = \ell$ for some $v \in Y$. If $v \notin Y'$ and $v \notin K$, then by Equation (8), $\ell = I_v[i, D] = \mathcal{A}[i - 1, (Y \setminus \{v\}, Y', \mathcal{P}', Y_{\text{sp}}, K')]$. By the induction hypothesis, there is a locally rooted subgraph F that is of type $D' = (Y \setminus \{v\}, Y', \mathcal{P}', Y_{\text{sp}}, K')$ and $\text{recdist}(F) \leq \ell$. Since D and D' are satisfying types, conditions (a) through (d) in the definition of satisfying types imply that F is of type D as well. If $v \in Y'$, then $\ell = I_v[i, D] = \mathcal{A}[i - 1, D'' = (Y \setminus \{v\}, Y' \setminus \{v\}, \mathcal{P} \setminus \{v\}, Y_{\text{sp}} \setminus \{v\}, K' \setminus \{v\})]$. By the induction hypothesis, there is a locally rooted subgraph F such that $\text{recdist}(F) \leq \ell$ and F is of type D''. This implies that the graph $F' = F \cup (\{v\}, \emptyset)$ is of type D and $\text{recdist}(F') = \text{recdist}(F) \leq \ell$.

Case 3. Suppose $\mathcal{A}[i, D] = I_{uv}[i, D] = \ell$ for some $u, v \in Y$ such that $uv \in E(G)$. If u and v are in the same block of $\mathcal{P}$, then $\mathcal{A}[i, D] = \mathcal{A}[i - 1, D] = \ell$. By the induction hypothesis, there is a locally rooted subgraph F that is of type D and $\text{recdist}(F) \leq \ell$. However, suppose u and v are in different blocks of $\mathcal{P}$. If $\mathcal{A}[i, D] = \mathcal{A}[i - 1, D] = \ell$, then again by the induction hypothesis, we have a locally rooted subgraph of type D and weight at most ℓ. Otherwise, $\mathcal{A}[i, D] = \mathcal{A}[i - 1, (Y, Y', \mathcal{P}', Y'_{\text{sp}}, K')] + \text{recdist}(uv)$ for a pair $(\mathcal{P}', Y'_{\text{sp}}) \in \mathbb{P}$. By the induction hypothesis, there is a locally rooted subgraph F' with type $D' = (Y, Y', \mathcal{P}', Y'_{\text{sp}}, K')$, such that $\text{recdist}(F') \leq \ell - \text{recdist}(uv)$ and there is no $u - v$ path in F'. By definition of pairs in $\mathbb{P}$ and transitivity of the shortness property, the graph $F = F' \cup (\{u, v\}, \{uv\})$ is a locally rooted subgraph that has type D, by satisfying all of properties (a) through (d). Since $\text{recdist}(F) \leq \ell$ we are done.

Case 4. When $\mathcal{A}[i, D] = F_w[i, D] = \ell$, for some $w \in V(G)$, then the arguments are similar to case 1.

Case 5. Suppose $\mathcal{A}[i, D] = \mathcal{A}[i-1, (Y, Y', \mathcal{P}_1, Y^1_{\text{sp}}, K'_1)] + \mathcal{A}[i-1, (Y, Y', \mathcal{P}_2, Y^2_{\text{sp}}, K'_2)] = \ell$, for a tuple $(\mathcal{P}_1, \mathcal{P}_2, Y^1_{\text{sp}}, Y^2_{\text{sp}}) \in \mathbb{Q}$ and for $K'_1 \cup K'_2 = K'$. By the induction hypothesis, for $j \in [2]$, there is a locally rooted subgraph F_j of type $D_j = (Y, Y', \mathcal{P}_j, Y^j_{\text{sp}}, K'_j)$. By the last two properties of the tuple $(\mathcal{P}_1, \mathcal{P}_2, Y^1_{\text{sp}}, Y^2_{\text{sp}})$, if F_1 and F_2 are forests, then $F_1 \cup F_2$ is also a forest. In addition, by transitivity of the shortness property, it follows that $F = F_1 \cup F_2$ is a locally rooted subgraph of type D. Since, $\text{recdist}(F) \leq \text{recdist}(F_1) + \text{recdist}(F_2) \leq \ell$, this proves the hypothesis. □

The next lemma links an optimal rectilinear Steiner arborescence to the values computed for the table $\mathcal{A}[,]$. First, we recall the definition of $c[,,]$. For a subset $X \subseteq X_t$, a partition $\mathcal{P}$ of X, and a set X_{sp} defined by selecting exactly one vertex from each block of $\mathcal{P}$, let $c[t, X, \mathcal{P}, X_{\text{sp}}]$ be the minimum weight of the subgraph F of G_t such that the following hold: (i) F has q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_j = X_t \cap V(C_j)$, (ii) $X_t \cap V(F) = X$, (iii) $K \cap V_t \subseteq V(F)$, and (iv) for each $j \in [q]$, and each vertex $w \in V(C_j) \setminus \{r_j\})$, the $w - r_j$ path in F is a shortest path in G and w has the shortness property with r_j. If there is no such subgraph F, then the value $c[t, X, \mathcal{P}, X_{\text{sp}}]$ is ∞.

Lemma A.4. *Let $\mathcal{T} = (\mathbb{T}, \{X_t\}_{t \in V(\mathbb{T})})$ be a nice tree decomposition of T. For a node t, let X_t be the corresponding bag, $X \subseteq X_t$, $\mathcal{P}$ be a partition of X, and X_{sp} be a set defined by selecting exactly one vertex from each block t of $\mathcal{P}$. Let V_t be the union of bags in the subtree rooted at t, and $K' = K \cap V_t$. Then $A[i, (X_t, X, \mathcal{P}, X_{\text{sp}}, K')] \leq c[t, X, \mathcal{P}, X_{\text{sp}}]$.*

Proof. Recall the nice tree decomposition $\mathcal{T} = (\mathbb{T}, \{X'_t\}_{t \in V(\mathbb{T})})$, of T, rooted at a node $\tilde{t}$. To each bag in $\mathcal{T}$, we add the root terminal r, thereby obtaining a new tree decomposition $\mathcal{T}' = (\mathbb{T}, \{X'_t\}_{t \in V(\mathbb{T})})$. The treewidth of the new tree decomposition is at most 1 more than that of the old tree decomposition. For the ease of presentation, we use $\mathcal{T} = (\mathbb{T}, \{X_t\}_{t \in V(\mathbb{T})})$ to denote the new tree decomposition of T. Note that all leaf bags and the root bag $X_{\tilde{t}}$, of $\mathcal{T}$, contain only one element r. As before, for any node $t \in V(\mathcal{T})$, the *level* of t is the height of the subtree rooted at t. Note that leaves in $\mathcal{T}$ other than root $\tilde{t}$ have level 1. The level of $\tilde{t}$ is the height of $\mathbb{T}$. By Proposition 2.1, the level of any node in $\mathbb{T}$ is at most N. For any node $t \in V(\mathcal{T})$, we use ℓ_t to denote the level of t. For any t, we denote the graph S_t as $(V_t, \{e \mid e \text{ is introduced in the subtree rooted at } t\})$, where V_t is the union of bags present in the subtree rooted at t.

We prove the following statement: for any $t \in V(\mathbb{T})$, $X \subseteq X_t$, a partition $\mathcal{P} = \{P_1, \ldots, P_q\}$ of X, and a set $X_{\text{sp}} = \{r_1, \ldots, r_q\}$ such that $r_j \in P_j$, it holds that $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\text{sp}}, K \cap V_t)] \leq c[t, X, \mathcal{P}, X_{\text{sp}}]$. We prove the statement using induction on the level of the node t. The base case is when $\ell_t = 1$. In this case, $X = \{r\}$. If $X = \{r\}$ and $\mathcal{P} = \{\{r\}\}$, by definition $\mathcal{A}[1, (X_t, X, \{X\}, X, K \cap V_t)] = 0 = c[t, X, \{X\}, X]$. Otherwise, $\mathcal{A}[1, (X_t, X, \mathcal{P}, X, K \cap V_t)] = \infty = c[t, X, \{X\}, X]$. Let t be a node in $V(\mathcal{T})$, $X \subseteq X_t$, $\mathcal{P}$ be a partition of X, X_{sp} be a set defined by selecting exactly one vertex from each block of $\mathcal{P}$, and $1 < \ell_t \leq N$. Let $K' = K \cap V_t$. If $(X_t \setminus X) \cap K \neq \emptyset$, then by definition $c[t, X, \mathcal{P}, X_{\text{sp}}] = \infty$ and so the statement holds. Suppose $(X_t \setminus X) \cap K = \emptyset$. Since $\widehat{T}$ is a Steiner arborescence for K, $K \subseteq V(\widehat{T})$. Since X_t is a bag in the tree decomposition $\mathcal{T}$, each terminal in a connected component C of $\widehat{T} - X_t$ is either fully contained in V_t or none of the terminals in the component C are present in V_t. Thus, there exists a set of connected components $C_1, \ldots, C_{\hat{j}}$ of $\widehat{T} - X_t$ such that $K' = K \cap V_t = T \cap (X_t \cup \bigcup_{j=1}^{\hat{j}} V(C_j))$. This implies that $(X_t, X, \mathcal{P}, X_{\text{sp}}, K')$ is a type. Let $\mathcal{P} = \{P_1, \ldots, P_q\}$, $X_{\text{sp}} = \{r_1, \ldots, r_q\}$ such that $r_j \in P_j$, and let F be a witness subgraph for the value $c[t, X, \mathcal{P}, X_{\text{sp}}]$. In other words, $\text{recdist}(F) = c[t, X, \mathcal{P}, X_{\text{sp}}]$ and the following conditions hold: (i) F has q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_j = X_t \cap V(C_j)$, (ii) $X_t \cap V(F) = X$, (iii) $K \cap V_t \subseteq V(F)$, and (iv) for each $j \in [q]$, and each vertex $w \in V(C_j) \setminus \{r_j\})$, the $w - r_j$ path in

F is a shortest path in G and w has the shortness property with r_j. We look at cases based on the nature of the node t.

Case 1: t is an introduce vertex node. Let t' be the child of t and $\{v\} = X_t \setminus X'_t$. Note that the level of t' is $\ell_t - 1$. If $v \notin V(F)$, then $c[t', X, \mathcal{P}, X_{\mathrm{sp}}] \leq \mathrm{recdist}(F)$. By Equations (8) and (12), $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathrm{sp}}, K')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, X_{\mathrm{sp}}, K')]$. By the induction hypothesis,

$$\mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, X_{\mathrm{sp}}, K')] \leq c[t', X, \mathcal{P}, X_{\mathrm{sp}}] \leq \mathrm{recdist}(F) = c[t, X, \mathcal{P}, X_{\mathrm{sp}}].$$

If $v \in V(F)$, then v appears as an isolated vertex in F, because v is an isolated vertex in S_t. By definition of X_{sp}, v must belong to X_{sp}. This implies that $c[t', X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, X_{\mathrm{sp}} \setminus \{v\}] \leq \mathrm{recdist}(F)$. By Equations (8) and (12),

$$\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathrm{sp}}, K')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, X_{\mathrm{sp}} \setminus \{v\}, K')].$$

By the induction hypothesis, $\mathcal{A}[\ell_t - 1, (X'_t, X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, X_{\mathrm{sp}} \setminus \{v\}, K')] \leq c[t', X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, X_{\mathrm{sp}} \setminus \{v\}] \leq \mathrm{recdist}(F \setminus \{v\}) = \mathrm{recdist}(F) = c[t, X, \mathcal{P}, X_{\mathrm{sp}}]$.

Case 2: t is an introduce edge node. Let t be labeled with the edge uv and t' be the child of t. In other words, $\{u, v\} \subseteq X_{t'} = X_t$. Note that the level of t' is $\ell_t - 1$. If $uv \notin E(F)$, then $c[t', X, \mathcal{P}, X_{\mathrm{sp}}] \leq \mathrm{recdist}(F)$. By Equations (9) and (12), we know that

$$\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathrm{sp}}, K')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, X_{\mathrm{sp}}, K')].$$

By the induction hypothesis,

$$\mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, X_{\mathrm{sp}}, K')] \leq c[t', X, \mathcal{P}, X_{\mathrm{sp}}] \leq \mathrm{recdist}(F) = c[t, X, \mathcal{P}, X_{\mathrm{sp}}].$$

Suppose $uv \in E(F)$. This means that there is a single component C, in F, that contains u, v. Let $r_C = C \cap X_{\mathrm{sp}}$. Then, for each vertex $w \in C$, w has the shortness property with r_C. Let $C'_1, \ldots, C'_{q'}$ be the connected components of $F - uv$. Since each component of F is a tree, the component C breaks into two components, C' and C'', of $F \setminus \{uv\}$. Without loss of generality, assume that $r_C, u \in C'$ and $v \in C''$. Notice that any vertex of C'' has the shortness property with v, and any vertex of C' continues to have the shortness property with r_C. Consider the partition $\mathcal{P}'$ to be $\{V(C'_1) \cap X, \ldots, V(C'_{q'}) \cap X\}$ and $X'_{\mathrm{sp}} = X_{\mathrm{sp}} \cup \{v\}$. This implies that $c[t', X, \mathcal{P}', X'_{\mathrm{sp}}] \leq \mathrm{recdist}(F \setminus \{uv\}) = \mathrm{recdist}(F) - \mathrm{recdist}(uv)$. By the induction hypothesis,

$$\mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}', X'_{\mathrm{sp}}, K')] \leq c[t', X, \mathcal{P}', X_{\mathrm{sp}}] \leq \mathrm{recdist}(F) - \mathrm{recdist}(uv).$$

The property of F implies that in the partition $\mathcal{P}'$, if we merge the blocks containing u and v, then we get the partition $\mathcal{P}$. This, along with the definition of X'_{sp}, implies that the tuple $(\mathcal{P}', X'_{\mathrm{sp}}) \in \mathbb{P}$. Thus, by Equations (9) and (12),

$$\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathrm{sp}}, K')] \leq \mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}', X'_{\mathrm{sp}}, K')] + \mathrm{recdist}(uv) \leq \mathrm{recdist}(F).$$

Case 3: t is a forget node. Let t' be the child of t and $\{w\} = X_{t'} \setminus X_t$. Note that the level of t' is $\ell_t - 1$. If $w \notin V(F)$, then $c[t', X, \mathcal{P}, X_{\mathrm{sp}}] \leq \mathrm{recdist}(F)$. By the induction hypothesis, $\mathcal{A}[\ell_t, (X_{t'}, X, \mathcal{P}, X_{\mathrm{sp}}, K')] \leq c[t', X, \mathcal{P}, X_{\mathrm{sp}}] \leq \mathrm{recdist}(F)$. By Equations (10) and (12), it is true that $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathrm{sp}}, K')] \leq \mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}, X_{\mathrm{sp}}, K')] \leq \mathrm{recdist}(F)$.

Suppose $w \in V(F)$. Note that w does not belong to X. Consider the set $X \cup \{w\} \subseteq X_{t'}$. Let C_j be the component of F containing w. Note that $P_j = V(C_j) \cap X$. Since, $w \notin X$, $w \neq C_j \cap X_{\mathrm{sp}}$. Let $\mathcal{P}'$ is a partition obtained from $\mathcal{P}$, by adding w to the block P_j. Then $\mathcal{P}'$ is a partition of $X \cup \{w\}$. This implies that $c[t', X \cup \{w\}, \mathcal{P}', X_{\mathrm{sp}}] \leq \mathrm{recdist}(F)$. By the induction hypothesis,

$$\mathcal{A}[\ell_{t'}, (X_{t'}, X \cup \{w\}, \mathcal{P}', X_{\mathrm{sp}}, K')] \leq c[t', X \cup \{w\}, \mathcal{P}', X_{\mathrm{sp}}] \leq \mathrm{recdist}(F).$$

By Equations (10) and (12),

$$\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\text{sp}}, K')] \leq \mathcal{A}[\ell_t - 1, (X_{t'}, X \cup \{w\}, \mathcal{P}', X_{\text{sp}}, K')] \leq \text{recdist}(F).$$

Case 4: t is a join node. Let t_1 and t_2 be the children of t. Here, $X_t = X_{t_1} = X_{t_2}$ and the level of $X_{t_j}, j \in \{1, 2\}$, is at most $\ell_t - 1$. Note that at least one of the children must have a level exactly $\ell_t - 1$. Without loss of generality, let X_{t_1} be that child. It is possible that the level of X_{t_2} is ℓ_{t_2}, which is less than $\ell_t - 1$. Let F_1 be the graph with vertex set $V(F) \cap V_{t_1}$ and edge set $E(F) \cap E(S_{t_1})$. Let F_2 be the graph with vertex set $V(F) \cap V_{t_2}$ and edge set $E(F) \setminus E(F_1)$. As F was a forest, the graphs F_1 and F_2 are also forests. Note that $F = F_1 \cup F_2$. Let $T_1' = V(F_1) \cap T$ and $T_2' = V(F_2) \cap T$. Since all connected components in $V(F)$ contain at least one vertex from X and X_t is a bag in the tree decomposition, all connected components in F_1 and F_2 contain at least one vertex from X. Let $C_1', \ldots, C_{q'}$ be the connected components of F_1 and $C_1'', \ldots, C_{q''}''$ be the connected components in F_2. Let $\mathcal{P}_1 = \{X \cap V(C_1'), \ldots, X \cap V(C_{q'}')\}$ and $\mathcal{P}_2 = \{X \cap V(C_1''), \ldots, X \cap V(C_{q''}'')\}$. Consider a new block C', from one of the partitions $\mathcal{P}_1$ or $\mathcal{P}_2$, and assume that $C' \subseteq C \in \mathcal{P}$. Let $r_{C'}$ be the unique vertex, in $V(C')$, that has minimum distance to the vertex $r_C \in C \cap X_{\text{sp}}$. Then, each vertex in $V(C)$ has the shortness property with $r_{C'}$. This way, we obtain, for each $j \in [2]$, a set X_{sp}^j defined by taking exactly one vertex from each block of $\mathcal{P}_j$. Thus, $c[t_1, X, \mathcal{P}_1, X_{\text{sp}}^1] \leq \text{recdist}(F_1)$ and $c[t_2, X, \mathcal{P}_2, X_{\text{sp}}^2] \leq \text{recdist}(F_2)$. By the induction hypothesis, $\mathcal{A}[\ell_{t_j}, (X_{t_j}, X, \mathcal{P}_j, X_{\text{sp}}^j, K_j')] \leq c[t_j, X, \mathcal{P}_j, X_{\text{sp}}^j]$ for $j \in \{1, 2\}$. The definitions of F, F_1, and F_2 imply that $\mathcal{P} = \mathcal{P}_1 \sqcup \mathcal{P}_2$. In addition, notice that the tuple $(\mathcal{P}_1, \mathcal{P}_2, X_{\text{sp}}^1, X_{\text{sp}}^2) \in \mathbb{Q}$. By Equations (11) and (12), $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\text{sp}}, K')] \leq \mathcal{A}[\ell_t - 1, (X_t, X, \mathcal{P}_1, X_{\text{sp}}^1, K_1')] + \mathcal{A}[\ell_{t_2}, (X_t, X, \mathcal{P}_2, X_{\text{sp}}^2, K_2')] \leq \text{recdist}(F_1) + \text{recdist}(F_2) = \text{recdist}(F)$. This concludes the proof. □

Finally, we prove Theorem 4.5 by showing that Rectilinear Steiner Arborescence can be solved in time $2^{O(\sqrt{n}\log n)} n^{O(1)}$.

Proof of Theorem 4.5. We take as input a set K of n terminal points, the Hanan grid G of K, and the weight function recdist. Furthermore, $r \in K$ is the root terminal. Then using Lemma 4.1, we compute a shortest path RSA $\widehat{T}$. By Lemma 4.4, we know that there is an optimal Steiner tree T_{opt} with $\text{tw}(\widehat{T} \cup T_{\text{opt}}) \leq 41\sqrt{n}$. In addition, from Proposition 2.1, we know that the height of this tree decomposition is at most $3(|V(G)| + |E(G)|)$. Based on the shortest path RSA $\widehat{T}$, we apply Lemma A.2, to enumerate all possible types D of G. We fix an integer $N = 3(|V(G)| + |E(G)|)$. For each $i \in [N]$ and each type D, the algorithm computes values $\mathcal{A}[i, D]$, according to Equations (7) and (12). The values in $\mathcal{A}[,]$ are filled in the increasing order of i. Finally, the algorithm outputs $\min_{i \in [N]} \mathcal{A}[i, (\{r\}, \{r\}, \{\{r\}\}, \{r\}, K)]$.

For the hypothetical subgraph T, fix an optimal nice tree decomposition $\mathcal{T}$, rooted at node $\tilde{t}$. Add the root terminal r to each bag of the nice tree decomposition (as described in the proof of Lemma A.4). The treewidth of this tree decomposition is at most $41\sqrt{n} + 1$. Let t be a node in the tree decomposition, of level ℓ_t. Let X_t be the bag of t and V_t be the union of bags in the subtree rooted at t. Let $K' = K \cap V_t$. Suppose $X \subseteq X_t$, $\mathcal{P}$ is a partition of X, and the set X_{sp} is defined by selecting exactly one vertex from each block of $\mathcal{P}$. By definition, $c[t, X, \mathcal{P}, X_{\text{sp}}]$ is the size of a locally rooted subgraph of type $(X_t, X, \mathcal{P}, X_{\text{sp}}, K')$. Then, Lemmas A.3 and A.4 imply that $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\text{sp}}, K')] = c[t, X, \mathcal{P}, X_{\text{sp}}]$. In particular, for the root $\tilde{t}$ of the tree decomposition, $\mathcal{A}[\ell_{\tilde{t}}, (\{r\}, \{r\}, \{\{r\}\}, \{r\}, K)] = c[\tilde{t}, \{r\}, \{\{r\}\}, \{r\}]$. Notice that $c[\tilde{t}, \{r\}, \{\{r\}\}, \{r\}]$ is the size of a minimum rectilinear Steiner arborescence of G.

However, by Lemma A.3, for all $i \in [N]$, if $A[i, (\{r\}, \{r\}, \{\{r\}\}, \{r\}, K)] = \ell$, then there is a locally rooted subgraph F that connects all terminals of K and satisfies $\text{recdist}(F) \leq \ell$. By description, the output of the algorithm is $\min_{i \in [N]} \mathcal{A}[i, (\{r\}, \{r\}, \{\{r\}\}, \{r\}, K)]$. Therefore, it must output the

weight of a minimum rectilinear Steiner arborescence of G. This proves the correctness of the algorithm.

The size of the table $\mathcal{A}[,]$ is $N \cdot 2^{O(\sqrt{n}\log n)}$, and each entry can be filled in time $2^{O(\sqrt{n}\log n)}n^{O(1)}$. Thus, the running time of the algorithm is $2^{O(\sqrt{n}\log n)}n^{O(1)}$. Using standard back-tracking tricks, we can also output an optimal RSA. This concludes the proof. □

ACKNOWLEDGMENTS

We would like to thank Professor Dániel Marx for helpful comments. We would also like to thank anonymous reviewers for their inciteful comments.

REFERENCES

[1] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. 2007. Fourier meets Möbius: Fast subset convolution. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC'07)*. ACM, New York, NY, 67–74.

[2] Marcus Brazil and Martin Zachariasen. 2015. *Optimal Interconnection Trees in the Plane: Theory, Algorithms and Applications*. Springer.

[3] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer-Verlag.

[4] Linda L. Deneen, Gary M. Shute, and Clark D. Thomborson. 1994. A probably fast, provably optimal algorithm for rectilinear Steiner trees. *Random Structures and Algorithms* 5, 4 (1994), 535–557.

[5] Stuart E. Dreyfus and Robert A. Wagner. 1971. The Steiner problem in graphs. *Networks* 1, 3 (1971), 195–207.

[6] Bernhard Fuchs, Walter Kern, Daniel Mölle, Stefan Richter, Peter Rossmanith, and Xinhui Wang. 2007. Dynamic programming for minimum Steiner trees. *Theory of Computing Systems* 41, 3 (2007), 493–500.

[7] Joseph L. Ganley. 1999. Computing optimal rectilinear Steiner trees: A survey and experimental evaluation. *Discrete Applied Mathematics* 90, 1–3 (1999), 161–171.

[8] Joseph L. Ganley and James P. Cohoon. 1997. Improved computation of optimal rectilinear Steiner minimal trees. *International Journal of Computational Geometry* 7, 5 (1997), 457–472. DOI: https://doi.org/10.1142/S0218195997000272

[9] M. R. Garey and D. S. Johnson. 1977. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics* 32, 4 (1977), 826–834.

[10] Qian-Ping Gu and Hisao Tamaki. 2012. Improved bounds on the planar branchwidth with respect to the largest grid minor size. *Algorithmica* 64, 3 (2012), 416–453.

[11] M. Hanan. 1966. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics* 14 (1966), 255–265.

[12] Frank K. Hwang. 1976. On Steiner minimal trees with rectilinear distance. *SIAM Journal on Applied Mathematics* 30, 1 (1976), 104–114.

[13] Frank K. Hwang, Dana S. Richards, and Pawel Winter. 1992. *The Steiner Tree Problem*. Annals of Discrete Mathematics, Vol. 53. North-Holland, Amsterdam, Netherlands.

[14] Philip N. Klein and Dániel Marx. 2014. A subexponential parameterized algorithm for Subset TSP on planar graphs. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*. 1812–1830.

[15] Ton Kloks. 1994. *Treewidth: Computations and Approximations*. Vol. 842. Springer Science & Business Media.

[16] L. Nastansky, S. M. Selkow, and N. F. Stewart. 1974. Cost-minimal trees in directed acyclic graphs. *Zeitschrift fur Operations Research* 18, 1 (1974), A59–A67.

[17] Jesper Nederlof. 2013. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica* 65, 4 (2013), 868–884.

[18] Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. 2013. Subexponential-time parameterized algorithm for Steiner tree on planar graphs. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS'13)*. Leibniz International Proceedings in Informatics, Vol. 20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 353–364.

[19] Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. 2014. Network sparsification for Steiner problems on planar and bounded-genus graphs. In *Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS'14)*. IEEE, Los Alamitos, CA, 276–285.

[20] Hans Jürgen Prömel and Angelika Steger. 2002. *The Steiner Tree Problem*. Friedr. Vieweg & Sohn, Braunschweig, Germany.

[21] Weiping Shi and Chen Su. 2005. The rectilinear Steiner arborescence problem is NP-complete. *SIAM Journal on Computing* 35, 3 (2005), 729–740. DOI: https://doi.org/10.1137/S0097539704371353

[22] Warren D. Smith and Nicholas C. Wormald. 1998. Geometric separator theorems and applications. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS'98)*.

[23] Clark D. Thomborson, Linda L. Deneen, and Gary M. Shute. 1987. Computing a rectilinear Steiner minimal tree in $n^{O(\sqrt{n})}$ time. In *Parallel Algorithms and Architectures*. Springer, 176–183.

Received April 2017; revised April 2019; accepted November 2019