Subexponential algorithms for rectilinear Steiner tree and arborescence problems

Fedor V. Fomin* Sudeshna Kolay† Daniel Lokshtanov* Fahad Panolan†
Saket Saurabh*†

Abstract

A *rectilinear Steiner tree* for a set T of points in the plane is a tree which connects T using horizontal and vertical lines, In the Rectilinear Steiner Tree problem, input is a set T of n points in the Euclidean plane ($\mathbb{R}^2$) and the goal is to find an rectilinear Steiner tree for T of smallest possible total length. A *rectilinear Steiner arborecence* for a set T of points and root $r \in T$ is a rectilinear Steiner tree S for T such that the path in S from r to any point $t \in T$ is a shortest path. In the Rectilinear Steiner Arborescence problem the input is a set T of n points in $\mathbb{R}^2$, and a root $r \in T$, the task is to find an rectilinear Steiner arborescence for T, rooted at r of smallest possible total length. In this paper, we give the *first* subexponential time algorithms for both problems. Our algorithms are deterministic and run in $2^{\mathcal{O}(\sqrt{n}\log n)}$ time.

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 $T \subseteq V(G)$. The task is to find a minimum-weight connected subgraph of G, which is a tree, containing all terminal nodes T. Steiner Tree is one of the central and best-studied problems in Computer Science, we refer to the books of Hwang, Richards, and Winter [13] and Prömel and Steger [19] for thorough introductions to the problem.

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

For the purposes of this paper, 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 (Hanan grid [11])**.** *Given a set T of n terminal points in the Euclidean plane $\mathbb{R}^2$, the* Hanan grid *G of T 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 x-axis) and a vertical line (line parallel to y-axis) through each point of T. 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;*

*University of Bergen.{fedor.fomin|daniello}@ii.uib.no
†The Institute of Mathematical Sciences {skolay|fahad|saket}@imsc.res.in

the weight of edge uv is the rectilinear distance between u and v. For a Hanan grid G we define a weight function $\mathsf{recdist}_G$ *from the edge set $E(G)$ to $\mathbb{R}$ such that for an edge $uv \in E(G)$,* $\mathsf{recdist}_G(uv) = d_1(u,v)$. *If the graph G is clear from the context we drop the subscript from* $\mathsf{recdist}_G$ *and only use* $\mathsf{recdist}$.

Let us note that when G is the Hanan grid of a set T of n points, then $T \subseteq V(G)$, $|V(G)| \leq n^2$, 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 T of n terminal points, the Hanan grid G of T and $\mathsf{recdist}_G$.
Output: A minimum Steiner tree for T in G.

Previous work on Rectilinear Steiner Tree. Though 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, Richards, and Winter [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 $3^n \cdot \log W \cdot |V(G)|^{\mathcal{O}(1)}$, where W is the maximum edge weight in G. For Rectilinear Steiner Tree, an adaptation of Dreyfus-Wagner algorithm provides an algorithm of running time $\mathcal{O}(n^2 \cdot 3^n)$. The survey of Ganley [7] summarizes the chain of improvements based on this approach concluding by the $\mathcal{O}(n^2 \cdot 2.62^n)$-time algorithm of Ganley and Cohoon [8]. Thobmorson et al. [22] and Deneen et al. in [4] gave randomized algorithms with running time $2^{\mathcal{O}(\sqrt{n}\log n)}$ for the special case of Rectilinear Steiner Tree when the terminal points T are drawn from a uniform distribution on a rectangle.

It is also worth to mention relevant parameterized algorithms for Steiner Tree on general graphs. Fuchs et al. [6] provides an algorithm with running time $\mathcal{O}((2+\varepsilon)^n |V(G)|^{f(1/\varepsilon)} \log W)$. Björklund et al. [1] and Nederlof [16] gave $2^n |V(G)|^{\mathcal{O}(1)} \cdot W$ time algorithms for Steiner Tree. Let us remark that since the distances between adjacent vertices in Hanan grid can be exponential in n, the algorithms of Björklund et al. and of Nederlof do not outperform the running time of Dreyfus-Wagner for the Rectilinear Steiner Tree problem. Interesting recent developments also concern Steiner Tree on planar graphs, and more generally, on graphs of bounded genus. While the existence of algorithms running in time subexponential in the number of terminals on these graph classes is still open, Pilipczuk et al. [17, 18] showed that Steiner Tree can be solved in time subexponential in the size of the Steiner tree on graphs of bounded genus.

In spite of the long history of research on Rectilinear Steiner Tree and Steiner Tree, the question whether Rectilinear Steiner Tree can be solved in time subexponential in the number of terminals remained open. In this paper we give the first such algorithm. The running time of our algorithm is $2^{\mathcal{O}(\sqrt{n}\log n)}$. Further, our techniques also yield the first subexponential algorithm for the related Rectilinear Steiner Arborescence problem.

Definition 2. *Let G be a graph, $T \subseteq V(G)$ a set of terminals, and $r \in T$ be a root vertex. A* Steiner arborescence of T in G *is a subtree $H \subseteq G$ rooted at r with the following properties:*

- *H contains all vertices of T, and*
- *For every vertex $t \in T \setminus \{r\}$, the unique path in H connecting r and t is also the shortest r–t path in G.*

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

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

Rectilinear Steiner Arborescence was introduced by Nastansky, Selkow, and Stewart [15] in 1974. Interestingly, the complexity of the problem was open until 2005, when Shu 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 Hwang's theorem [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. While our algorithm follows this general scheme, the implementations of both steps for our problems are entirely different from [14].

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 $\widehat{S}$ of $T = \{t_1, \ldots, t_n\}$, we implement the following greedy strategy. We build $\widehat{S}$ starting from vertex t_1 and gradually connect new terminals to the tree. When we connect terminal t_{i+1} to tree S_i spanning the first i terminals, we select a shortest monotone (containing at most one "bend") path from t_{i+1} to S_i in the Hanan grid. If there are two such paths, we select one of them according to the structure of S_i. The constructed tree $\widehat{S}$ can be seen as a "shortest path" rectilinear Steiner tree. The property of $\widehat{S}$ which is crucial for the algorithm is that there is an optimal Steiner tree S_{opt} such that graph $S = \widehat{S} \cup S_{\mathsf{opt}}$ is of treewidth $\mathcal{O}(\sqrt{n})$.

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

Let us recall 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, see e.g. [3, Theorem 7.8]. 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; in particular 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, i.e., 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 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 S. We guess some structural information about the virtual tree decomposition $\mathcal{T}$ of S. 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 S. 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. So the final step of the algorithm involves investigating all the 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 S and therefore of G.

2 Preliminaries

For a positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For 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 if 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 $\mathcal{P}$, that 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 $\mathcal{P}$, by $\mathcal{P} \setminus B$ 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 a vertex, denoted as $\mathsf{degree}_G(v)$, 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, v is denoted as uv. A path, where vertices $\{u_1, u_2, \ldots, u_\ell\}$ appear in sequence, is denoted by $u_1 u_2 \cdots u_\ell$. 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 internal. 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 weight function $w : E(G) \to \mathbb{R}$, for a subgraph H of G, we use 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 the 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 $V(G_1 \cup G_2) = V(G_1) \cup V(G_2)$, while the edge set $E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$.

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 $\max_{t \in V(\mathbb{T})} |X_t| - 1$ and the *treewidth* of G is the minimum width over all tree decompositions of G and is denoted by $\mathsf{tw}(G)$. A tree decomposition $(\mathbb{T}, \mathcal{X})$

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:

1. **Introduce node**: a node t that has only one child t' where $X_t \supset X_{t'}$ and $|X_t| = |X_{t'}| + 1$.
2. **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.
3. **Forget vertex node**: a node t that has only one child t' where $X_t \subset X_{t'}$ and $|X_t| = |X_{t'}| - 1$.
4. **Join node**: a node t with two children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$.
5. **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. One can show that a tree decomposition of width t can be transformed into a nice tree decomposition of the same width t and with $\mathcal{O}(t|V(G)|)$ nodes, see e.g. [3].

2.1 Planar graph embeddings and minors

A graph is *planar* if can be embedded in the plane. That is, 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 $\Pi(v)$, such that 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 $v \in V(G)$, $\Pi(v) \notin \Pi(e)$. Now we define the notion of a minor of a graph G.

Definition 3. *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 edge deletions and contractions.*

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

Definition 4. *Let G and H be two connected graphs, and $H \leq_m G$. A* minor model *or simply a* model *of a 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.*

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

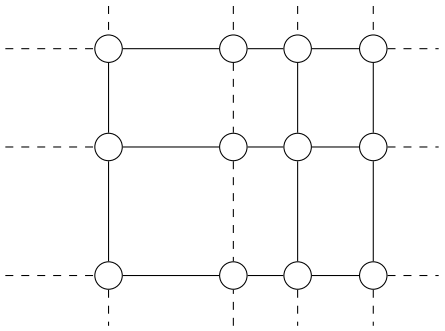

Figure 1: The solid edges define a subgrid of a grid

Definition 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 a column of the grid G. The vertices in the first row, n-th row, the first column and m-th columns are called the* boundary vertices *of the grid. The vertices that are not boundary vertices are called* internal vertices*.*

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 internal 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 1).

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

Definition 6. *Let G be a planar graph with a planar embedding Π, and 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 if it lies in the internal region of C, and external if it lies in the external region of C. An edge in $E(G)$ is called an external edge if there is at least one point on its curve that lies in the external region. It is called an internal edge if there is at least one point on its curve that lies in the internal region.*

By definition of a planar embedding, an edge of $V(G)$ can be exactly one of the three kinds: an edge of C, and external edge or an internal edge. Similarly a vertex can be exactly one of the three kinds: a vertex of C, an external vertex or an internal vertex.

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 $\mathcal{P}(H) = \{C_v | v \in V(H)\}$ 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' by embedding derived *from Π.*

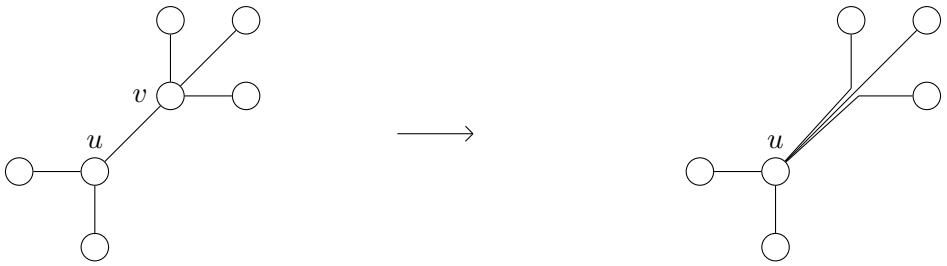

Figure 2: Derived embedding: the edge uv is contracted to the vertex u.

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:

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

This ensures the two properties above. □

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

Lemma 1. *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 that 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 $u \in H'$ 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 $u' \in V(C_u), w' \in V(C_w)$. 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 graph C_u of G. There are at most two vertices, u_1 and u_2, that are incident with marked edges in $V(C_u)$. Since $G[C_u]$ is a connected graph, there is a a shortest path $P_{u_1u_2}$ connecting u_1 and u_2 in $G[C_u]$. We call $P_{u_1v_2}$ as 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 $u \in C'$.

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 4, there is a vertex $u \in H$ such that $w \in C_u$. If $u \in H'$, 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 $u \notin H'$, $C_u \cap C = \emptyset$. However, 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 internal vertex $v \in H$, 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 internal vertex $v \in H$ 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 C_v is connected, to maintain planarity, it follows that C_v is in the internal region of C.

Lastly, we show that for the internal vertex $v \in H$, 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 $u \in H'$, let D_u denote the restriction of C_u in G'. Since, for the internal vertex $v \in H'$, C_v is completely contained in the internal region of C, $D_v = C_v$. There are four neighbors of $v \in H'$ 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 three 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 the shortest path between w and x_3 be Q. If $w = x_3$, then w must be internal in P_{12} and thus has at least two neighbors in C_v. Since w is also a marked vertex, it has a third neighbor outside C_v, thereby making its degree in G at least three. Otherwise, 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 three 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 three in G. This completes the proof. □

Finally, our proof will be using the following version of the fundamental result of Robertson et al. [20].

Proposition 1 ([10]). *Let t be a nonnegative integer. Then every planar graph G of treewidth at least $9t/2$ contains a $t \times t$ grid as a minor.*

2.2 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 the horizontal edges as well as all the vertical edges in R are contiguous.

The following observation contains some simple facts about monotone paths (see Figure 3).

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

(a) There is at least one and at most 2 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 the 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.

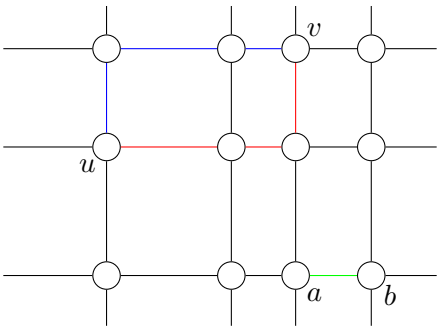

Figure 3: The red and blue paths are the two monotone u–v paths, and the green path is the single monotone a–b path.

(c) If there are two monotone paths between u and v, then one path has a horizontal edge incident with u and other path has a vertical edge incident with u

Definition 7. *Suppose we are given a Hanan grid G of a set of terminals T 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.*

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 T be an input set of terminals (points in $\mathbb{R}^2$), $|T| = n$, and G be the Hanan grid of T. Furthermore, let $\mathsf{recdist}_G$ denote the weight function on the edge set $E(G)$. For brevity we will use $\mathsf{recdist}$ for $\mathsf{recdist}_G$. Our algorithm is based on a dynamic programming over vertex subsets of size $\mathcal{O}(\sqrt{n})$ 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 $\widehat{S}$, there is an optimal Steiner tree S_{opt} such that $\widehat{S} \cup S_{\mathsf{opt}}$ has bounded treewidth. Finally, keeping a hypothetical tree decomposition of $\widehat{S} \cup S_{\mathsf{opt}}$ 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 $T = \{t_1, \ldots, t_n\}$ 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 T. We define a shortest path RST $\widehat{S}$ through the following constructive greedy process. Initially, we set S_1 to the graph $(\{t_1\}, \emptyset)$, which is a rectilinear

Steiner tree of $\{t_1\}$. In the i^{th} step, we compute a rectilinear Steiner tree S_{i+1} of $\{t_1, \ldots t_{i+1}\}$ from S_i as follows. If $t_{i+1} \in V(S_i)$, then we set $S_{i+1} = S_i$. Otherwise, let v be a vertex in S_i such that $\mathsf{recdist}(v, t_{i+1}) = \min_{u \in V(S_i)} \mathsf{recdist}(u, t_{i+1})$. If there is only one monotone t–v path, then let Q be this path. Otherwise there are two monotone t–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 S_i which is incident with v, then we choose Q to be the monotone t–v path such that the edge in Q incident with v is a horizontal edge. Otherwise we choose Q to be the monotone t–v path such that the edge in Q incident with v is a vertical edge. Then we construct S_{i+1} by adding a monotone path Q to S_i. After $n-1$ iterations, we construct a tree $\widehat{S} = S_n$ of G, which is a Steiner tree of T. This is our shortest path RST.

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

Lemma 2. *Given a set T of terminal points and the Hanan grid G of T, a shortest path RST $\widehat{S}$ of T can be constructed in polynomial time.*

Proof. Consider the procedure used to define a shortest path RST. The procedure involves $|T|$ steps. In each step, a shortest path, between two vertices of G, is found out. Since the shortest path subroutine can be executed in polynomial time, and there are polynomially many calls to this subroutine, the construction of $\widehat{S}$ requires polynomial time. □

Next we give an upper bound on the number of bend vertices in a shortest path RST.

Lemma 3. *The number of bend vertices in $\widehat{S}$ is at most n.*

Proof. We prove the assertion by induction on the number of iterations to construct the solution $\widehat{S}$. Towards this, using induction on i, we prove that the number of bend vertices in S_i is at most i. In the base case, S_1 is a single vertex with no bend vertices. Suppose that the statement holds for S_{i-1}. If t_i is already contained in S_{i-1}, then $S_i = S_{i-1}$. By induction, the number of bend vertices in S_{i-1} is at most $i-1$, and therefore the number of bend vertices in S_i is at most i. Otherwise, we find a vertex $v \in V(\widehat{S}_{i-1})$ such that a shortest path Q between t_i and $\widehat{S}_{i-1}$ ends with v. Since Q is a shortest path between t_i and $\widehat{S}_{i-1}$, the set of internal vertices of Q is disjoint from $V(\widehat{S}_{i-1})$. By induction hypothesis, S_{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 S_{i-1} then the number of bend vertices in S_i is at most $i-1+1 = i$. By the way we define shortest path RST, if v is not a bend vertex in S_{i-1}, it is also not bend in S_i. Therefore in this case the number of bend vertices in S_i is also at most i. This concludes the proof. □

3.2 Supergraph of an optimal RST with bounded treewidth

In this section 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 $\widehat{S}$, we show the existence of an optimal Steiner tree S_{opt} such that the treewidth of $\widehat{S} \cup S_{\mathsf{opt}}$ is sublinear in the number of terminal points T. First, we show that there is an optimal Steiner tree in G that has a bounded number of bends.

Lemma 4. *Let T be a set of n points in $\mathbb{R}^2$ and G be the Hanan grid of T. Then there is an optimal rectilinear Steiner tree of T such that the number of bend vertices in T is at most $3n$.*

Proof. We prove the lemma using induction on $n = |T|$. The base case is when $n = 1$. Since the tree $(T, \emptyset)$ is an optimal Steiner tree when $|T| = 1$, the number of bend vertices in the tree $(T, \emptyset)$ 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 one and the hypothesis is still true.

Consider the induction step where $n = |T| > 2$. Let S_{opt} be an optimal Steiner tree of T in G. Since S_{opt} is an optimal Steiner tree of $|T|$, there are at least 2 leaves and each leaf node is a terminal. Also, there is a pair $\{t_1, t_2\}$ of leaf terminals such that, in the t_1–t_2 path P of S_{opt}, there is exactly one internal vertex u with degree at least three in S_{opt}. This means that all other internal vertices in P are of degree exactly two in S_{opt}. Consider the sub-paths P_1 and P_2, which are t_1–u and t_2–u paths. Suppose there is a terminal t appearing as an internal vertex on P. Without loss of generality, we can assume that $t \in V(P_1)$ and t is the second terminal vertex in the path P_1 (first terminal vertex is t_1). Let us denote the t_1–t sub-path of P_1 as P_3. By definition, all the internal vertices between t_1 and t are degree two non-terminal vertices. Let S_1 be the tree obtained by deleting $V(P_3) \setminus \{t\}$ from S_{opt}. Since S_{opt} is an optimal Steiner tree of T, S_1 is an optimal Steiner tree of $T \setminus \{t_1\}$ and P_3 is a minimum weight t_1–t path. Since $|T \setminus \{t_1\}| = n - 1$, by induction hypothesis, there is an optimal Steiner tree S' of $T \setminus \{t_1\}$ such that the number of bend vertices is at most $3(n-1)$. Let P_3' be a monotone t_1–t path. Since $\mathsf{recdist}(S' \cup P_3') \leq \mathsf{recdist}(S_1 \cup P_3) = \mathsf{recdist}(S_{\mathsf{opt}})$, we have that $S' \cup P_3'$ is an optimal Steiner tree of T. 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 $S' \cup P_3'$ is a bend vertex in S', or a bend vertex in P_3', or $b = t$. This implies that the number of bend vertices in $S' \cup P_3'$ is at most $3(n-1) + 1 + 1 \leq 3n$.

The remaining case is that the t_1–t_2 path P has exactly one internal vertex u of degree at least three, while all other internal vertices are of degree two and are not terminals. Let S_1 be the tree obtained by deleting $V(P_1 \cup P_3) \setminus \{u\}$ from S_{opt}. Since S_{opt} is an optimal Steiner tree of T, S_1 is an optimal Steiner tree of $T \setminus \{t_1, t_2\} \cup \{u\}$. Again, the sub-paths P_1 and P_2 must be minimum weight t_1–u and t_2–u paths respectively. Since $|T \setminus \{t_1, t_2\} \cup \{u\}| = n-1$, by induction hypothesis, there is an optimal Steiner tree S' of $T \setminus \{t_1, t_2\} \cup \{u\}$ such that the number of bend vertices is at most $3(n-1)$. By optimality of S_{opt}, for any minimum weight t_1–u path P_1' and t_2–u path P_2', $\mathsf{recdist}(S' \cup P_1' \cup P_2') = \mathsf{recdist}(S_{\mathsf{opt}})$. Thus, we have that $S = S' \cup P_1' \cup P_2'$ is an optimal Steiner tree of T. 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 one each. This means that u could be a new bend vertex in S, 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 three. Thus, the total number of bend vertices in S is at most $3n$. This completes the proof. □

With respect to a shortest path RST $\widehat{S}$, of G, we prove the next Lemma. In particular we show that the treewidth of $S = \widehat{S} \cup S_{\mathsf{opt}}$ is at most $36\sqrt{n}$. Here, S_{opt} is a carefully chosen optimal Steiner tree for T. In order to get the desired upper bound on the treewidth of S, we show that it does not contain $\mathcal{O}(\sqrt{n}) \times \mathcal{O}(\sqrt{n})$ grid as a minor. Towards this we prove that if there is a large grid then we can find a "clean part of the grid" (subgrid not containing vertices of T and bend vertices of either $\widehat{S}$ or S_{opt}) and reroute some of the paths in either $\widehat{S}$ or S_{opt}, that contradicts either the way $\widehat{S}$ is constructed or the optimality of S_{opt}.

Lemma 5. *Given a set T of n points and a shortest path RST $\widehat{S}$ of T, there is an optimal rectilinear Steiner tree S_{opt} of T with the property that the treewidth of $\widehat{S} \cup S_{\mathsf{opt}}$ is bounded by $36\sqrt{n}$.*

Proof. Among the optimal Steiner trees of T with the minimum number of bend vertices, we select a tree S_{opt} which has maximum edge intersection with $E(\widehat{S})$. From Lemma 4, it follows that the number of bend vertices in S_{opt} is at most $3n$.

Let $S = \widehat{S} \cup S_{\mathsf{opt}}$. Let $\widehat{B}$ and B_{opt} be the set of bend vertices in $\widehat{S}$ and S_{opt} respectively. Let $U = T \cup \widehat{B} \cup B_{\mathsf{opt}}$ and $N = V(G) \setminus U$. Since $|T| = n$, $|\widehat{B}| \leq n$, and $|B_{\mathsf{opt}}| \leq 3n$, $|U| \leq 5n$. Let Π_S be a planar embedding of S, obtained by deleting all the edges and vertices not in S from the planar embedding Π of G. We show that the treewidth of S is at most $36\sqrt{n}$. For the sake of contradiction, assume that $\mathsf{tw}(S) > 36\sqrt{n}$. Then, by Proposition 1, there is a $8\sqrt{n} \times 8\sqrt{n}$ grid

H appearing as a minor of S. 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 $\widehat{S}$ or S_{opt}). Since $|U| \leq 5n$, the number of marked vertices in H is at most $5n$. Since H is a $8\sqrt{n} \times 8\sqrt{n}$ grid, there are at least $6n$ vertex disjoint 3×3 subgrids in H. 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) $\mathsf{degree}_{\widehat{S}}(w), \mathsf{degree}_{S_{\mathsf{opt}}}(w) \in \{0, 2\}$. *If for any $S_i \in \{\widehat{S}, S_{\mathsf{opt}}\}$, $\mathsf{degree}_{S_i}(w) = 2$, then the two edges in S_i incident with w are of same kind (either horizontal or vertical).*

(ii) If one horizontal (vertical) edge incident with w is present in S, then the other horizontal (vertical) edge incident with w is also present in S. Hence $\mathsf{degree}_S(w) \in \{2, 4\}$.

Note that H' is a connected graph and is a minor of a connected graph S. Let $\Pi_{H'}$ be a planar embedding derived from Π_S. By Lemma 1, we know that there is a simple cycle C' in S with the following properties.

(i) $V(C') \subseteq \bigcup_{u \in V(H')} C_u$.

(ii) 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 the vertices of $V(S) \setminus \bigcup_{v \in V(H')} C_v$ (which includes U) are not in the internal region of C'.

(iii) For the internal vertex $v \in V(H')$, all the vertices in C_v are in the internal region of C'.

(iv) Finally, there is a vertex $w \in C_v$, in the internal region of C', such that $\mathsf{degree}_S(w) \geq 3$. By Observation 6, $\mathsf{degree}_S(w) = 4$.

That is, there is a cycle C' in the Hanan grid G such that $V(C') \subseteq V(S) \setminus U$, 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 S, which is in the internal region of C'. The following claim follows from Observation 6.

Claim 1. *Let $u, v \in V(G)$ such that the points u and v are 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 $v_1v_2 \in E(S)$ on the line L, then all the edges on the line segment L belong to $E(S)$.*

Let C' be a minimum-weight cycle satisfying properties $(i), (ii)$ and (iv) and w' be a vertex of degree 4 in the internal region of C'. Let $E_{w'}$ be the set of edges of S 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, $V(G') \subseteq V(G) \setminus U$, $E(G') \subseteq E(S)$, and all the subdivision vertices in G' have degree exactly 2 in S.*

Proof. The definition of G' implies that $V(G') \subseteq V(G) \setminus U$ and $E(G') \subseteq E(S)$. 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 S on the line L_1 such that $l_x < w'_x, r_x > w'_x$, and the distances $\mathsf{recdist}(w', l)$ and $\mathsf{recdist}(w', r)$ are minimized. Similarly, let a, b be degree 4 vertices of S on the line L_2 such that $a_y > w'_y, b_y < w'_y$ and the distances $\mathsf{recdist}(w', a)$ and $\mathsf{recdist}(w', b)$ are minimized. Let $R = \{a_x, w'_x, b_x\}$ and $C = \{l_y, w'_y, r_y\}$. 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 that **(a)** there is no edge $uv \in E(S)$ such that uv is in the internal

region of C'' and $uv \in E(S) \setminus E(G'')$. Suppose not. Among all such edges, let uv be an edge such that $\mathsf{recdist}(w', u)$ 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 $E(S)$. 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 S. This edge belongs to $E(S) \setminus E(G'')$, 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 z. First, suppose $u \neq z$. Let the line segment L' of L connect u and z. As shown above, u either belongs to C', or is in the internal region of C'. Following from Observation 6, since uv is an edge of S, 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 $u = z$, 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 as well as the two edges of L_i, incident on u, belong to S and are not in the external region of C'. Hence, there are both horizontal and vertical edges in S which are incident with u. Since u is not an external vertex of C', the degree of u in S 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.

The condition (a) implies that the internal region of C'' does not have an edge of S. Since all the 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'. Also, all the edges in C'' are not in the outer region of C'. Since degree of l, r, a and b in S are 4, by Claim 1, all the edges in C'' belong to $E(S)$. Since w' is in the internal region of C'', condition (iv) holds for C''. Also the vertices and edges of C'' either belong to C' or are in the internal region of C'. Thus conditions (i) and (ii) also hold. Then, by the minimality of C', $C'' = C'$. Since the degree of w' in S is 4, by Claim 1, all the edges in $E_{w'}$ belong to $E(S)$.

Now we need to show that all the subdivision vertices of G' have degree 2 in S. Suppose not. Let u be a subdivision vertex in G' such that degree of u in S is greater than 2. By Observation 6, degree of u in S 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 S. □

The next claim provides us with the insight on how subpaths of $\widehat{S}$ and S_{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. *$F_h \subseteq E(\widehat{S})$ and $F_v \subseteq E(S_{\mathsf{opt}})$.*
2. *$F_v \subseteq E(\widehat{S})$ and $F_h \subseteq E(S_{\mathsf{opt}})$.*

Proof. Let $G_1 = G'[E(\widehat{S})]$ and $G_2 = G'[E(S_{\mathsf{opt}})]$. 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 as well as a horizontal edge. But 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 $E(G_1) - E(D)$. Also, the edges of D are of a different orientation than the edge e. That is, if all the edges of D are horizontal, then e is a vertical edge, and vice-versa. As shown above, all the 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 that 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 $E(\widehat{S})$. Traversing from u along Q_{uw}, let e^* be the first edge of $E(\widehat{S})$ that is encountered. If e^* is a vertical edge, then (C_1, e^*) is a pair which satisfies the above description. However, the distance between C_1 and e^* is strictly smaller than the distance between C_1 and uv, which is a contradiction. On the other hand, suppose that e^* is a horizontal edge and that 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 $E(S_{\mathsf{opt}})$ and not to $E(\widehat{S})$. Moreover, all the edges of Q_{uw} must belong to a single component C_2 of G_2. This implies that all the 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 $E(C_1) \subseteq E(\widehat{S})$ and $e_w \notin E(\widehat{S})$, 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 $e \in E(\widehat{S})$ while $e_u \notin E(\widehat{S})$. 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. □

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 9 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 S_{opt} and the vertical paths belong to $\widehat{S}$. For a path P_{ijk}, we use P_{ij} and P_{jk} to denote the sub-paths of P_{ijk} connecting u_i and u_j, and u_j and u_k respectively. Let the length of the sub-path P_{12} be l_1 and the length of the sub-path P_{23} be l_2. By the definition of G', the length of P_{45} is also l_1, and the length of P_{56} is l_2. Let the length of P_{14} be p. The length of both P_{25} and P_{36} is p. (See Figure 4).

Suppose $l_1 + l_2 > 2p$. Then we consider the graph S^* formed by deleting in S_{opt} the path P_{123} and adding the two paths P_{14} and P_{36}. Since all the subdivision vertices of G' are of degree 2 in S, we have that S^* is a Steiner tree of weight strictly less than the weight of S_{opt}. This contradicts the choice of S_{opt}. Hence, this is not possible.

Suppose $l_1 + l_2 \leq 2p$. Without loss of generality, let $l_1 \leq l_2$. Thus, $l_1 \leq p$. Consider the two paths P_{147} and P_{258}. They are vertical paths of $\widehat{S}$, such that all the vertices in these paths belong to $V(G) \setminus U$ (that is, non-terminals and non-bend vertices) and have degree 2 in $\widehat{S}$ (by Observation 6). This implies that all the edges in any path $R \in \{P_{147}, P_{258}\}$ are added in a single step while constructing $\widehat{S}$. Since both the paths are parallel, by Observation 4, if a path R_1 in $\widehat{S}$ has both P_{147} and P_{258} as sub-paths, then R cannot be a shortest path between its endpoints. Thus, by construction of $\widehat{S}$, both P_{147} and P_{258} could not have been added to $\widehat{S}$ in a single step of the construction. Also by construction, one of them is added to $\widehat{S}$ before the other. Without loss of generality, let P_{147} be added before P_{258}. Again by construction, a path,

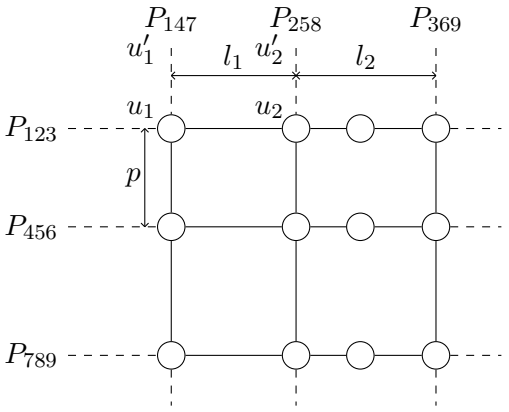

Figure 4: The subgrid G'

containing P_{258} as a sub-path, was added in the i^{th} step, to connect a terminal t to the already constructed $\widehat{S}_{i-1}$. Let R^* be the path added to S_{i-1}. By definition of G', this terminal t must lie outside the region formed by the subgrid G'. Since P_{258} was part of a shortest path between $\widehat{S}_{i-1}$ and t, by Observation 4, t must lie on a row strictly higher than or strictly lower than the rows in G'. Suppose that this terminal lies above P_{123}. By Observation 6, both the vertical edges incident on u_1 belong to $\widehat{S}$. Since u_1 and u_2 are of degree 2 in $\widehat{S}$, both the 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 $\widehat{S}$. This implies that both the vertical edges incident with u_1 are present in S_{i-1}. Let $u_1u'_1$ be the vertical edge present in S_{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 t, obtained by concatenating the horizontal path between u'_1 and u'_2, and the sub-path of R^* connecting u'_2 and t. The length of the path R_2 is strictly less than that of R^* and $u'_1 \in S_{i-1}$. This is a contradiction. The case when t lies below any row in G' is identical to the other case.

Thus, there is no such subgrid G' of G, such that G' is a subgraph of S. This implies that there is no $8\sqrt{n} \times 8\sqrt{n}$ grid as a minor in S. Due to Proposition 1, the treewidth of S must be at most $\frac{9}{2} \cdot 8\sqrt{n} = 36\sqrt{n}$. This completes the proof. □

3.3 Dynamic Programming Algorithm for Rectilinear Steiner Tree

In this section we utilize all the results proved in the previous sections and design our algorithm for Rectilinear Steiner Tree. By Lemma 5, we known that given a shortest path RST $\widehat{S}$, there exists an optimum Steiner tree S_{opt} such that the treewidth of $S = \widehat{S} \cup S_{\mathsf{opt}}$ is bounded by $36\sqrt{n}$. The idea of the algorithm is to *implicitly* do a dynamic programming over a tree decomposition of S, *even though we do not know what S is*, to compute an optimum Steiner tree for T.

Suppose we know the subgraph S of G such that there is an optimum Steiner tree fully contained in S. Then we can do the well known algorithm for Steiner tree over the tree decomposition of S [3]. To give a proper intuition to our algorithm, we first recall the important step of the dynamic programming algorithm for Steiner tree over the tree decomposition of the graph S (see [3, Theorem 7.8] for more details). Let $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ be a nice tree decomposition of S where $\mathbb{T}$ is a rooted tree. For a node t, let V_t be the union of all the bags present in the subtree of $\mathbb{T}$ rooted at t. For a node t, we define a graph $S_t = (V_t, E_t = \{e \in E(S) \ : \ e \text{ is introduced in the subtree rooted at } t\})$. The important step in the algorithm for Steiner Tree is to compute the following information: for each bag X_t, $X \subseteq X_t$ and a partition $\mathcal{P} = (P_1, \ldots, P_q)$ of X, 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 $i \in [q]$. That is, $\mathcal{P}$ corresponds to connected components of F.
2. $X_t \cap V(F) = X$. That is, the vertices of $X_t \setminus X$ are untouched by F.
3. $T \cap V_t \subseteq V(F)$. That is, all the terminal vertices in S_t belong to F.

For our purposes the second step is redundant but we still carry it to give a proper analogy with the known algorithm for Steiner Tree over graph of bounded treewidth we are referring to. Note that the number of blocks in the partition $\mathcal{P}$ is q. Throughout this section, q will denote the number of blocks of the partition in question.

It is known that computing the values $c[t, X, \mathcal{P}]$ for each tuple $(t, X, \mathcal{P})$, where $t \in V(\mathbb{T}), X \subseteq X_t$ and $\mathcal{P}$ is a partition of X, is enough to compute the value of an optimum Steiner tree of T in S and this can be computed in time $\mathsf{tw}^{\mathcal{O}(\mathsf{tw})}|V(S)|^{\mathcal{O}(1)}$ (See chapter 7 in the book [3]). In our case, we do not know the graph $S = \widehat{S} \cup S_{\mathsf{opt}}$ and a tree decomposition of S, but we know that the treewidth of S is at most $36\sqrt{n}$. This implies that the number of choices for bags in a tree decomposition of S is bounded by $n^{\mathcal{O}(\sqrt{n})}$. Consider the properties 1 and 2 mentioned above. They are *local* properties of the witness subgraph F with respect to the bag X_t. But the property 3 says that all the terminals in the subgraph S_t should be present in F. In fact we can bound the *potential* sets $T \cap V(S_t)$ using the rectilinear Steiner tree $\widehat{S}$. Observe that any bag X_t is a separator of size $\mathcal{O}(\sqrt{n})$ for S and thus for $\widehat{S}$, which implies that *for every connected component C of $\widehat{S} - X_t$, either $T \cap V(C)$ is fully in V_t or no vertex in $T \cap V(C)$ 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 $\widehat{S} - X_t$ is at most $4|X_t| \leq 144(\sqrt{n} + 1)$. As observed before, for any connected component C of $\widehat{S} - X_t$, either $T \cap V(C)$ is fully in V_t or no vertex in $T \cap V(C)$ belongs to V_t. This implies that the potential sets $T' \subseteq T$ such that $T' = (V_t \setminus X_t) \cap T$ is bounded by $2^{144(\sqrt{n}+1)}$ and we can enumerate them in sub-exponential time. Using this observation we could keep track of property 3 as well, even though we do not know the graph S and its tree decomposition.

Now, we move towards the formal explanation of our algorithm. Towards that we first define a notion of *type* which is the analogue of a tuple $(t, X, \mathcal{P})$ in the normal dynamic programming we explained above.

Definition 8. *A type is a tuple $(Y, Y' \subseteq Y, \mathcal{P}, T')$ such that the following holds.*

(i) Y is a subset of $V(G)$ of size at most $36\sqrt{n} + 2$.
(ii) $\mathcal{P}$ is a partition of Y'.
(iii) There exists a set of components $C_1, \ldots, C_q$ in $\widehat{S} - Y$ such that $T' = T \cap (Y' \cup \bigcup_{i=1}^{q} V(C_i))$.

Informally, in a type $(Y, Y', \mathcal{P}, T')$, Y represents a potential bag Y of a node (say t) in a tree decomposition of S. The set Y' and partition $\mathcal{P}$ have the same meaning as that of $(t, Y', \mathcal{P})$ in the normal dynamic programming algorithm for Steiner Tree. The set T' is the set of terminals in the graph S_t. The next lemma gives an upper bound on the number of types.

Lemma 6. *There is a $2^{\mathcal{O}(\sqrt{n}\log n)}n^{\mathcal{O}(1)}$ time algorithm $\mathcal{B}$ enumerating all the types.*

Proof. We know that type is a tuple $(Y, Y', \mathcal{P}, T')$ satisfying properties $(i)-(iii)$. Since $|V(G)| \leq n^2$, the number of choices for Y is $n^{\mathcal{O}(\sqrt{n})}$. The cardinality of Y is at most $36\sqrt{n}+2$, thus for a fixed Y, the number of choices for Y' is $2^{\mathcal{O}(\sqrt{n})}$ and the number of choices for the partition $\mathcal{P}$, of Y', is $n^{\mathcal{O}(\sqrt{n})}$. By definition of the Hanan grid G, each vertex in G has at most 4 neighbors. Thus, $\widehat{S} \setminus Y$ has at most $4(36\sqrt{n}+2)$ components. Hence, on fixing Y, the choices for T' is at most $2^{\mathcal{O}(\sqrt{n})}$. Thus we get an upper bound of $2^{\mathcal{O}(\sqrt{n}\log n)}$ on the number of types. Furthermore, it can be enumerated in time $2^{\mathcal{O}(\sqrt{n}\log n)}n^{\mathcal{O}(1)}$. □

Our algorithm is a dynamic programming algorithm over the types of S. As motivated earlier, this algorithm essentially describes the ideas of the dynamic programming algorithm for Steiner Tree over a tree decomposition of an input graph. 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 the information that is necessary for the correctness of the dynamic programming algorithm for Steiner Tree over a tree decomposition of S. 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 S. For ease of understanding, let it be a nice tree decomposition and let the tree be rooted at a node $r \in \mathbb{T}$. 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. 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 $(V_t, \{e \mid e \text{ is introduced in a the subtree rooted at } t\})$. Let $T' = V_t \cap T$. Then, for any $X \subseteq X_t$, and a partition $\mathcal{P}$ of X having q blocks, $\mathcal{A}[i, (X_t, X, \mathcal{P}, T')] = c[t, X, \mathcal{P}]$. 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 $\emptyset \neq P_i = X_t \cap V(C_i)$, (ii) $X_t \cap V(F) = X$, and (iii) $T \cap V_t \subseteq V(F)$. 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 above subset of pairs (i, D). Of course we do not know S and thus we do not know the tree decomposition $\mathcal{T}$, so we store values in the table $\mathcal{A}[,]$ in such a way that given *any* nice tree decomposition of S, we have information pertaining to it. Thus, given a pair (i, D) where $D = (Y, Y' \subseteq Y, \mathcal{P}, T')$, we view Y as a bag of some hypothetical nice tree decomposition, $\mathcal{T}$, of S 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. We guess the relationship between a bag at level i and its children, which must be at level $i - 1$. 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 $\mathcal{A}[i, (X_t, X, \mathcal{P}, T')]$ using that values of $\mathcal{A}$ calculated at step $i - 1$ of the algorithm. The calculation ensures that $\mathcal{A}[i, (X_t, X, \mathcal{P}, T')] = c[t, X, \mathcal{P}]$. 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 t^* in T

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

To define $\mathcal{A}[i, D]$ for $i \in [N] \setminus \{1\}$ and a type $D = (Y, Y', \mathcal{P}, T')$, 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 S and having level i. This viewpoint results in the following recurrence. For all $v \in Y$,

$$I_v[i, D] = \begin{cases} \infty & \text{if } v \notin Y' \text{ and } v \in T \\ \mathcal{A}[i-1, (Y \setminus \{v\}, Y', \mathcal{P}, T')] & \text{if } v \notin Y' \text{ and } v \notin T \\ \mathcal{A}[i-1, (Y \setminus \{v\}, Y' \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, T' \setminus \{v\})] & \text{if } v \in Y' \end{cases} \tag{2}$$

Intuitively, if Y is a bag corresponding to a node t in a tree decomposition of S and T' is the set of terminals in S_t, then Equation 2 corresponds to the computation of $c[t, Y', \mathcal{P}]$ in the dynamic programming algorithm of Steiner Tree. See [3, Theorem 7.8] for more detailed explanation.

For all $u, v \in Y$ and $uv \in E(G)$,

$$I_{uv}[i, D] = \min \left\{ \min_{\mathcal{P}'} \left\{ \mathcal{A}[i-1, (Y, Y', \mathcal{P}', T')] + \mathsf{recdist}(uv) \right\}, \mathcal{A}[i-1, D] \right\}, \tag{3}$$

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 $w \in V(G)$,

$$F_w[i,D] = \min\left\{\min_{\mathcal{P}'}\left\{\mathcal{A}[i-1,(Y\cup\{w\},Y'\cup\{w\},\mathcal{P}',T')]\right\}, \mathcal{A}[i-1,(Y\cup\{w\},Y',\mathcal{P},T')]\right\}, \tag{4}$$

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

$$J[i,D] = \min_{\substack{\mathcal{P}=\mathcal{P}_1\sqcup\mathcal{P}_2\\ T'=T'_1\cup T'_2}}\left\{\mathcal{A}[i-1,(Y,Y',\mathcal{P}_1,T'_1)] + \mathcal{A}[i-1,(Y,Y',\mathcal{P}_2,T'_2)]\right\} \tag{5}$$

Equation 5 corresponds to a computation in a join node.

We define $\mathcal{A}[i,D]$ for $i \in [N]\setminus\{1\}$ and type $D = (Y,Y',\mathcal{P},T')$ as,

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

For each $i \in [N]$ and each type D, we associate with $A[i,D]$ a subgraph of S. We say that a subgraph F is of type $(Y,Y',\mathcal{P},T')$, 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 $V(C_i)\cap Y = P_i$.
(b) $V(F)\cap T = T'$.

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

Lemma 7. *Let $i \in [N]$ and D be a type. Furthermore, let $A[i,D]$ be computed by the Equation 6, and have a finite value ℓ. Then there is a subgraph F, of type D, such that* $\mathsf{recdist}(F) \le \ell$.

Proof. We prove the statement using induction on i. Since the graph $(\{t^*\},\emptyset)$ is of type $(\{t^*\},\{t^*\},\{\{t^*\}\},\{t^*\})$, the base case holds trivially. Let $1 < i \le N$ and $D = (Y,Y',\mathcal{P},T')$ 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 $\mathsf{recdist}((F) \le \ell$. 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 $v \notin T$, then by Equation 2, $\ell = I_v[i,D] = \mathcal{A}[i-1,(Y\setminus\{v\},Y',\mathcal{P}',T')]$. By induction hypothesis there is a subgraph F of type $D' = (Y\setminus\{v\},Y',\mathcal{P}',T')$ and $\mathsf{recdist}(F) \le \ell$. The definition of satisfying type D or D' (conditions (a) and (b)) implies that F is of type D as well. If $v \in Y'$, then $\ell = I_v[i,D] = \mathcal{A}[i-1,D'' = (Y\setminus\{v\},Y'\setminus\{v\},\mathcal{P}\setminus\{v\},T'\setminus\{v\})]$. By induction hypothesis, there is a subgraph F such that $\mathsf{recdist}(F) \le \ell$ and F is of type D''. This implies that the graph $F' = F\cup(\{v\},\emptyset)$ is of type D and $\mathsf{recdist}(F') = \mathsf{recdist}(F) \le \ell$. In all other cases the proof follows by similar arguments. □

The next lemma helps us to 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 $\emptyset \neq P_i = X_t \cap V(C_i)$, (ii) $X_t \cap V(F) = X$, and (iii) $T \cap V_t \subseteq V(F)$. If there is no such subgraph F, then the value of $c[t, X, \mathcal{P}]$ is ∞.

Lemma 8. *Let $\mathcal{T} = (\mathbb{T}, \{X_t\}_{t \in V(\mathbb{T})})$ be a nice tree decomposition of S. 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 $T' = T \cap V_t$. Then $A[i, (X_t, X, \mathcal{P}, T')] \leq c[t, X, \mathcal{P}]$.*

Proof. Let $\mathcal{T} = (\mathbb{T}, \{X_t\}_{t \in V(\mathbb{T})})$ be a nice tree decomposition of S and $|V(\mathbb{T})| \leq 3(|V(S)| + |E(S)|) \leq N$. Recall that t^* is a fixed terminal. We add t^* to all the bags in $\mathcal{T}$. This new decomposition still satisfies all the property of a tree decomposition. The width of this new tree decomposition increases by at most 1. 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 S. Note that all the leaf bags and the root bag contain only one element, t^*. 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 $\mathbb{T}$. 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 $(V_t, \{e \mid e \text{ is introduced in a the subtree rooted at } t\})$, 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, $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, T \cap V_t)] \leq c[t, X, \mathcal{P}]$. We prove the statement using induction on the level of the node t. The base case is when $\ell_t = 1$. In this case $X_t = \{t^*\}$. If $X = \{t^*\}$ and $\mathcal{P} = \{\{t^*\}\}$, by definition $\mathcal{A}[1, (X_t, X, \{X\}, T \cap V_t)] = 0 = c[t, X, \{X\}]$. Otherwise $\mathcal{A}[1, (X_t, X, \mathcal{P}, T \cap V_t)] = \infty = c[t, X, \{X\}]$. 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 $T' = T \cap V_t$. If $(X_t \setminus X) \cap T \neq \emptyset$, then by definition $c[t, X, \mathcal{P}] = \infty$ and so the statement holds. Suppose $(X_t \setminus X) \cap T = \emptyset$. Since $\widehat{S}$ is a Steiner tree for T, $T \subseteq V(\widehat{S})$. Since X is a bag in the tree decomposition $\mathcal{T}$, all the terminals in a connected component C of $\widehat{S} - X_t$ 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 $C_1, \ldots, C_j$ of $\widehat{S} - X_t$ such that $T' = T \cap V_t = T \cap (X_t \cup \bigcup_{i=1}^{j} V(C_i))$. This implies that $(X_t, X, \mathcal{P}, T')$ is a type. Let $\mathcal{P} = \{P_1, \ldots, P_q\}$ and F be a witness subgraph for the value $c[t, X, \mathcal{P}]$. That is, $\mathsf{recdist}(F) = c[t, X, \mathcal{P}]$ and the following conditions hold: (i) F has q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_i = X_t \cap V(C_i)$, (ii) $X_t \cap V(F) = X$, and (iii) $T \cap V_t \subseteq V(F)$. We analyse 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 $c[t', X, \mathcal{P}] \leq \mathsf{recdist}(F)$. By Equations 6 and 2, $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, T')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, T')]$. By induction hypothesis, $\mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, T')] \leq c[t', X, \mathcal{P}] \leq \mathsf{recdist}(F) = c[t, X, \mathcal{P}]$. If $v \in V(F)$, then v appears as an isolated vertex in F, because v is an isolated vertex in S_t. This implies that $c[t', X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}] \leq \mathsf{recdist}(F)$. By Equations 6 and 2, $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, T')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, T')]$. By induction hypothesis, $\mathcal{A}[\ell_t - 1, (X'_t, X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, T')] \leq c[t', X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}] \leq \mathsf{recdist}(F \setminus \{v\}) = \mathsf{recdist}(F) = c[t, X, \mathcal{P}]$.

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

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 $c[t', X, \mathcal{P}'] \leq \mathsf{recdist}(F \setminus \{uv\}) = \mathsf{recdist}(F) - \mathsf{recdist}(uv)$. By induction hypothesis, $\mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}', T')] \leq$

$c[t', X, \mathcal{P}'] \leq \mathsf{recdist}(F) - \mathsf{recdist}(uv)$. 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 $\mathcal{P}$. Thus, by Equations 6 and 3, $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, T')] \leq \mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}', T')] + \mathsf{recdist}(uv) \leq \mathsf{recdist}(F)$.

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 $c[t', X, \mathcal{P}] \leq \mathsf{recdist}(F)$. By induction hypothesis, $\mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}, T)] \leq c[t', X, \mathcal{P}] \leq \mathsf{recdist}(F)$. By Equations 6 and 4, $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, T')] \leq \mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}, T')] \leq \mathsf{recdist}(F)$.

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

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 $X_{t_i}, i \in \{1, 2\}$, is $\ell_t - 1$. 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 $T'_1 = V(F_1) \cap T$ and $T'_2 = V(F_2) \cap T$. Since all the connected components in $V(F)$ contain at least one vertex from X and X_t is a bag in the tree decomposition, all the connected components in F_1 as well as in 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, $c[t_1, X, \mathcal{P}_1] \leq \mathsf{recdist}(F_1)$ and $c[t_2, X, \mathcal{P}_2] \leq \mathsf{recdist}(F_2)$. By induction hypothesis, $\mathcal{A}[\ell_{t_i}, (X_{t_i}, X, \mathcal{P}_i, T'_i)] \leq c[t_i, X, \mathcal{P}_i]$ for $i \in \{1, 2\}$. 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 $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, T')] \leq \mathcal{A}[\ell_t - 1, (X_t, X, \mathcal{P}_1, T'_1)] + \mathcal{A}[\ell_t - 1, (X_t, X, \mathcal{P}_2, T'_2)] \leq \mathsf{recdist}(F_1) + \mathsf{recdist}(F_2) = \mathsf{recdist}(F)$. □

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

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

Proof. We take as input a set T of n terminal points, the Hanan grid G of T and the weight function $\mathsf{recdist}$. Then using Lemma 2 we compute a shortest path RST $\widehat{S}$. By Lemma 5, we know that there is an optimal Steiner tree S_{opt} with $\mathsf{tw}(\widehat{S} \cup S_{\mathsf{opt}}) \leq 36\sqrt{n}$. Based on the shortest path RST $\widehat{S}$, we apply Lemma 6, to enumerate all possible types D of G. We fix an integer $N = 3(|V(G)| + |E(G)|)$ and a terminal t^* in T. 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 $\min_{i \in [N]} \mathcal{A}[i, (\{t^*\}, \{t^*\}, \{\{t^*\}\}, T)]$.

For the hypothetical set S, fix an optimal nice tree decomposition $\mathcal{T}$, rooted at node r. Add a fixed terminal t^* to each bag of the nice tree decomposition (As described in the proof of Lemma 8). The treewidth of this tree decomposition is bounded by $36\sqrt{n} + 1$. 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 $T' = T \cap V_t$. 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 $(X_t, X, \mathcal{P}, T')$. Then, Lemmata 7 and 8 imply that $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, T')] = c[t, X, \mathcal{P}]$. In particular, for the root r of the tree decomposition, $\mathcal{A}[\ell_r, (\{t^*\}, \{t^*\}, \{\{t^*\}\}, T)] = c[r, \{t^*\}, \{\{t^*\}\}]$. Notice, that $c[r, \{t^*\}, \{\{t^*\}\}]$ is the size of a minimum Steiner tree of G.

On the other hand, by Lemma 7, for all $i \in [N]$, if $A[i, (\{t^*\}, \{t^*\}, \{\{t^*\}\}, T)] = \ell$ then there is a subgraph F that connects all the terminals of T, and that satisfies $\mathsf{recdist}(F) \leq \ell$. As the algorithm outputs $\min_{i \in [N]} \mathcal{A}[i, (\{t^*\}, \{t^*\}, \{\{t^*\}\}, T)]$, 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 $N \cdot 2^{\mathcal{O}(\sqrt{n}\log n)}$ and each entry can be filled in time $2^{\mathcal{O}(\sqrt{n}\log n)} n^{\mathcal{O}(1)}$. Thus, the running time of the algorithm is $2^{\mathcal{O}(\sqrt{n}\log n)} n^{\mathcal{O}(1)}$. 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 are again given T, and the root vertex $r \in T$ as an input of Rectilinear Steiner Arborescence. Furthermore, let $|T| = n$ and G be the Hanan grid of T. We assume that the root vertex r is placed at $(0, 0)$ in $\mathbb{R}^2$. Let $\mathsf{recdist}_G$ be the weight function defined on the edges of G. In short, we use $\mathsf{recdist}$ for $\mathsf{recdist}_G$. Our aim is to design a subexponential algorithm for Rectilinear Steiner Arborescence. The algorithm is very similar to that for Rectilinear Steiner Tree. We define a rectilinear Steiner arborescence, called *shortest path RSA*, for a graph G. Then, for a shortest path RSA $\widehat{S}$, we show that there exists an optimal rectilinear Steiner arborescence S_{opt} such that the treewidth of $\widehat{S} \cup S_{\mathsf{opt}}$. Aided by this information, we design a 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*, which is a rectilinear Steiner Arborescence as follows.

We give an arbitrary ordering $\{r, t_1, \ldots t_k\}$ 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 S_1 to a r–t_1 monotone path. This is a rectilinear Steiner arborescence of $\{r, t_1\}$. In the i^{th} step, we compute a rectilinear Steiner arborescence S_{i+1} of $\{t_1, \ldots t_{i+1}\}$ from S_i as follows. If $t_{i+1} \in V(S_i)$, then we set $S_{i+1} = S_i$. Otherwise, for each vertex $u \in S_i$ let ℓ_u^1 be the length of a shortest u–t_{i+1} path. Let ℓ_u^2 be the length of the shortest r–u path that exists in S_i. Also, ℓ denotes the length of a shortest r–t_{i+1} path. Let $N \subseteq V(S_i)$ 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 t_{i+1}–u^* path, then let Q be that path. Otherwise there are two monotone t_{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 S_i which is incident with u^*, then we choose Q to be the monotone t_{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 t_{i+1}–u^* path such that the edge in Q incident with u^* is a vertical edge. Then, we construct S_{i+1} by adding the monotone path Q to S_i. After $n-1$ iterations, we construct a tree $\widehat{S} = S_n$ of G, which is a Steiner arborescence of T. This is our shortest path RSA.

It is possible to construct a shortest path RSA in polynomial time.

Lemma 9. *Given a set T of terminal points, and the Hanan grid G of T, a shortest path RSA $\widehat{S}$, of T, can be constructed in polynomial time.*

Similar to Lemma 3, we can give a bound on the bend vertices of a shortest path RSA.

Lemma 10. *The number of bend vertices in $\widehat{S}$ is at most n.*

4.2 Supergraph of an optimal RSA with bounded treewidth

Let T be an input set of points for Rectilinear Steiner Arborescence, $r \in T$ is a root terminal, and G is the Hanan grid of T. In this part, given a shortest path RSA $\widehat{S}$, we show the existence of an optimum rectilinear Steiner arborescence S_{opt} of T with the property that the

treewidth of $\widehat{S} \cup S_{\mathsf{opt}}$ is sublinear in the number of input points. Similar to Lemma 4, we can show that there is an optimal rectilinear Steiner arborescence with a bounded number of bend vertices.

Lemma 11. *Let T be a set of n terminals in $\mathbb{R}^2$, with a root terminal $r \in T$ and G be the Hanan grid of T. Then there is an optimum rectilinear Steiner arborescence of T in G such that the number of bend vertices in G is at most $3n$.*

After finding a shortest path RSA, as described by Lemma 9, we prove the following lemma.

Lemma 12. *Given a set T of n points, with $r \in T$ as the root terminal, and a shortest path RSA $\widehat{S}$ of T, there is an optimal rectilinear Steiner arborescence S_{opt} of T with the property that the treewidth of $\widehat{S} \cup S_{\mathsf{opt}}$ is bounded by $36\sqrt{n}$.*

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

We show that $\widehat{S} \cup S_{\mathsf{opt}}$ has $\mathcal{O}(\sqrt{n})$ treewidth. For the sake of contradiction, suppose $\widehat{S} \cup S_{\mathsf{opt}}$ has treewidth greater than $36\sqrt{n}$. Then, by Proposition 1, there is a $8\sqrt{n} \times 8\sqrt{n}$ grid H appearing as a minor in $\widehat{S} \cup S_{\mathsf{opt}}$. 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 $\widehat{S}$ or a bend vertex of S_{opt}, then we mark the vertex v in H. By Lemmata 10 and 11, the number of vertices of H that get marked is at most $5n$. By the arguments given in Lemma 5, 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 $\widehat{S}$ or S_{opt}.
2. There are four vertices $u_1, \ldots, u_4$ which are of degree exactly four in $\widehat{S} \cup S_{\mathsf{opt}}$. All other vertices are of degree exactly two in $\widehat{S} \cup S_{\mathsf{opt}}$.
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 the horizontal paths belong to S_{opt} and not $\widehat{S}$, while all the vertical paths belong to exactly $\widehat{S}$ and not S_{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 T be a set of terminals with the root terminal r placed at the origin. Let G be the Hanan grid of T and S_{mml} be an edge minimal Steiner arborescence for T. Let $u, v \in V(S_{\mathsf{mml}})$ 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 S_{mml}.*

Then it cannot be the case that all internal vertices of Q have degree two in S_{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 S_{mml} such that every internal vertex of Q is degree two in S_{mml}. Let S' be the graph obtained by deleting the internal vertices and edges of Q. Since, for each $t \in T$, there is a unique r–t path in S_{mml}, if t is still connected to r, in S', then the r–t path is still a shortest r–t path. We show that all

terminals are still connected to r in S'. This implies that S' is a Steiner arborescence, thereby contradicting the edge minimality of S_{mml}.

Suppose there is a terminal t, such that the r–t path Q' of S_{mml} had an edge in common with $E(Q)$. Since every internal vertex of Q is of degree two in S_{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 t. However, from Observation 5, it cannot be the case that Q is a shortest r–t path. Thus, for no terminal t, does the r–t path in S_{mml} intersect with Q. Thus, each terminal t remains connected to r in S'. Hence, we conclude that such a path Q cannot exist in an edge minimal Steiner arborescence S_{mml}. □

By definition, both $\widehat{S}$ and S_{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 $\mathbb{R}^2$. Also, without loss of generality, let the horizontal paths belong to S_{opt} and the vertical paths belong to $\widehat{S}$. 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 $l > p$. Then, we consider the Steiner arborescence formed by deleting, in S_{opt}, the path P_{12} and adding the path P_{24}. The resulting Steiner arborescence has weight strictly less than that of weight of S_{opt}. This contradicts the choice of S_{opt}. Hence, this is not possible.

Now, suppose $l \leq p$. Consider the two paths P_{13} and P_{24}. They are paths of $\widehat{S}$. 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 $\widehat{S}$ 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 t to the already constructed S_{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 S_{i-1} and t, t must lie on a row strictly higher than the rows of G'. Since, u_1 and u_2 are not bend vertices in $\widehat{S}$, they have neighbors u'_1 and u'_2 in $\widehat{S}$. Also, u'_1 and u'_2 are on the same row, and u'_1 is on the same column as u_1 while u'_2 is on the same column as u_2. Let $P_{u'_1}$ be the path from r to u'_1 in S_{i-1}, $P_{u'_2}$ be the subpath of P between r and u'_2, and P_t be the subpath of P between u'_2 and t. By definition of a shortest path, the subpath $P_{u'_2}$ is a shortest path between r and u'_2 and the subpath P_t is a shortest path between u'_2 and t. 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 $P_{u'_1} \cup P' \cup P_t$ is a shortest r–t path. Notice that, P is of weight at least (u'_2, u_2, u_4). Since $l \leq p$, this implies that P is of weight strictly more that the path P'. However, by the description of construction of $\widehat{S}$, the path $P' \cup P_t$ is a better candidate than the path P in step i. This contradicts the choice of adding the path P to form S_i. Therefore, it is not possible that $l \leq p$.

Thus we obtain a contradiction to the fact that $\widehat{S} \cup S_{\mathsf{opt}}$ has a grid minor greater than $8\sqrt{n}$. This proves that $\widehat{S} \cup S_{\mathsf{opt}}$ has treewidth less than $36\sqrt{n}$. □

4.3 Dynamic Programming Algorithm for Rectilinear Steiner Arborescence

Using the results of the previous sections we design a subexponential algorithm for Rectilinear Steiner Arborescence. By Lemma 12, we known that given a shortest path RSA $\widehat{S}$, there exists an optimal Steiner arborescence S_{opt} such that the treewidth of $S = \widehat{S} \cup S_{\mathsf{opt}}$ bounded by $36\sqrt{n}$. As in the case of the Rectilinear Steiner Tree problem, we do not know the graph S. However, we simulate a dynamic programming algorithm which contains all the information needed to compute an optimal Steiner arborescence on the tree decomposition of S.

Again, if we knew the subgraph S of G such that there is an optimal Steiner arborescence fully contained in S, we could design a dynamic programming algorithm for Rectilinear Steiner Arborescence over the tree decomposition of S. This algorithm is similar in ideas, to the algorithm for Steiner Tree over a tree decomposition of a given graph. Let $(\mathbb{T}, \mathcal{X}' = \{X'_t\}_{t \in V(\mathbb{T})})$ be a nice tree decomposition of S where $\mathbb{T}$ is a rooted tree. Let the root vertex be z. From this we obtain a tree decomposition $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ by adding the 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, i.e., 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.

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 $X_{\mathsf{sp}} = \{r_1, \ldots, r_q\}$ such that $r_i \in P_i$, for each $i \in [q]$, the value $c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$ 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 $i \in [q]$. That is, $\mathcal{P}$ corresponds to connected components of F.
2. $X_t \cap V(F) = X$. That is, the vertices of $X_t \setminus X$ are untouched by F.
3. $T \cap V_t \subseteq V(F)$. That is, all the terminal vertices in G_t belong to F.
4. For each $i \in [q]$, and each vertex $w \in V(C_i) - \{r_i\}$, 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 $c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$ for each tuple $(t, X, \mathcal{P}, X_{\mathsf{sp}})$, 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 $c[z, \{r\}, \{\{r\}\}, \{r\}]$ 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 T in S. In our case, we do not know the graph $S = \widehat{S} \cup S_{\mathsf{opt}}$ and a tree decomposition of S, but we know that the treewidth of S is at most $36\sqrt{n}$. This implies that the number of choices for bags in a tree decomposition of S is bounded by $n^{\mathcal{O}(\sqrt{n})}$. Notice that the properties 1–3 are same as the properties maintained for designing a dynamic programming algorithm for Rectilinear Steiner Tree on a tree decomposition of S. Property 4 is necessary to solve the Rectilinear Steiner Arborescence problem. Each vertex $r_i \in X_{\mathsf{sp}}$ can be thought of as a local root vertex for the component C_i of F. The intuition behind this property is as follows. Suppose F was a subgraph of an edge minimal Steiner arborescence S^* rooted at r. Also, for each $i \in [q]$, r_i is the unique vertex in C_i that has minimum distance to r in S. Then, for each $i \in [q], w \in C_i$, the w–r path in S^* satisfies Property 4. In particular, if S^* was an optimal Steiner arborescence, Property 4 still 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. That is, 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* which is the analogue of a tuple $(t, X, \mathcal{P}, X_{\mathsf{sp}})$ in the normal dynamic programming for Rectilinear Steiner Arborescence.

Definition 9. *A type is a tuple $(Y, Y' \subseteq Y, \mathcal{P}, Y_{\mathsf{sp}}, T')$ such that the following holds.*

(i) Y is a subset of $V(G)$ of size at most $36\sqrt{n}+2$.
(ii) $\mathcal{P}$ is a partition of Y'.
(iii) There exists a set of components $C_1, \ldots, C_q$ in $\widehat{S} \setminus Y$ such that $T' = T \cap (Y' \cup \bigcup_{i=1}^{q} V(C_i))$.
(iv) Y_{sp} has exactly one vertex r_i from each block $P_i \in \mathcal{P}$.

In a type $(Y, Y', \mathcal{P}, Y_{\mathsf{sp}}, T')$, Y represents a potential bag Y of a node (say t) in a tree decomposition of S. The set Y', partition $\mathcal{P}$ and Y_{sp} have the same meaning as that of $(t, Y', \mathcal{P}, Y_{\mathsf{sp}})$ in the normal dynamic programming algorithm for Rectilinear Steiner Arborescence over a tree decomposition of an input graph. The set T' is the set of terminals in the graph G_t. We can show that the number of types is bounded.

Lemma 13. *There is a $2^{\mathcal{O}(\sqrt{n}\log n)} n^{\mathcal{O}(1)}$ time algorithm $\mathcal{B}$ enumerating all the types.*

This proof is similar to the proof for Lemma 6 and hence we omit it here.

Below, we explain the steps of our algorithm. 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 the information that is necessary for the correctness of the dynamic programming algorithm for Steiner Arborescence over a tree decomposition of S. Since we do not know the graph S, we assume a hypothetical nice tree decomposition $\mathcal{T} = (\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ of S. We may also assume that the nice tree decomposition is rooted at a node $z \in \mathbb{T}$. The level of a vertex $t \in \mathbb{T}$ is the height of the subtree of $\mathbb{T}$ rooted at t. 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 $(V_t, \{e \mid e \text{ is introduced in a the subtree rooted at } t\})$. Let $T' = V_t \cap T$. 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 $\mathcal{P}$, $\mathcal{A}[i, (X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')] = c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$. As mentioned before, $c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$ 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 $\emptyset \neq P_i = X_t \cap V(C_i)$, (ii) $X_t \cap V(F) = X$, (iii) $T \cap V_t \subseteq V(F)$, and (iv) 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 w has the shortness property with r_i. 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 S is given to us, even though in reality we do not even know the graph S. Our dynamic programming is designed to mimic a dynamic programming algorithm for Steiner Arborescence over a tree decomposition of an input graph.

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.

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

In order to define $\mathcal{A}[i, D]$ for $i \in [N] \setminus \{1\}$ and a type $D = (Y, Y', \mathcal{P}, Y_{\mathsf{sp}}, T')$, we first define many intermediate values and take the minimum over all such values.

For all $v \in Y$,

$$I_v[i, D] = \begin{cases} \infty & \text{if } v \notin Y' \text{ and } v \in T \\ \infty & \text{if } v \in Y' \text{ but } \{v\} \notin \mathcal{P} \\ \mathcal{A}[i-1, (Y \setminus \{v\}, Y', \mathcal{P}, Y_{\mathsf{sp}}, T'] & \text{if } v \notin Y' \text{ and } v \notin T \\ \mathcal{A}[i-1, (Y \setminus \{v\}, Y' \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, Y_{\mathsf{sp}} \setminus \{v\}, T' \setminus \{v\})] & \text{if } v \in Y' \end{cases} \tag{8}$$

Intuitively, if Y is a bag corresponding to a node t in a tree decomposition of S and T' is the set of terminals in G_t, then Equation 8 corresponds to computation of $c[t, Y', \mathcal{P}, Y_{\mathsf{sp}}]$ in the dynamic programming algorithm 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 $P \cap Y_{\mathsf{sp}}$. We first define a set $\mathbb{P}$ of pairs $(\mathcal{P}', Y'_{\mathsf{sp}})$ on Y'. For a pair $(\mathcal{P}', Y'_{\mathsf{sp}})$, $\mathcal{P}'$ denotes a partition of Y', while Y'_{sp} is defined by taking exactly one vertex per block of $\mathcal{P}'$. For a pair $(\mathcal{P}', Y'_{\mathsf{sp}})$ 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 $v \in P' \cap Y'_{\mathsf{sp}}$, 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 $v \in P' \cap Y'_{\mathsf{sp}}$, every vertex of P' has the shortness property with v. Assume $u^* \in Y'_{\mathsf{sp}} \cap P_u$ and $v^* \in Y'_{\mathsf{sp}} \cap P_v$. 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 $u, v \in Y$,

$$I_{uv}[i, D] = \min \left\{ \min_{(\mathcal{P}', Y'_{\mathsf{sp}}) \in \mathbb{P}} \left\{ \mathcal{A}[i-1, (Y, Y', \mathcal{P}', Y'_{\mathsf{sp}}, T')] + \mathsf{recdist}(uv) \right\}, \mathcal{A}[i-1, D] \right\} \tag{9}$$

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 $w \in V(G)$,

$$\begin{aligned} F_w[i, D] \quad = \quad \min \Bigg\{ & \min_{\mathcal{P}'} \left\{ \mathcal{A}[i-1, (Y \cup \{w\}, Y' \cup \{w\}, \mathcal{P}', Y_{\mathsf{sp}}, T')] \right\}, \\ & \mathcal{A}[i-1, (Y \cup \{w\}, Y', \mathcal{P}, Y_{\mathsf{sp}}, T')] \Bigg\}, \end{aligned} \tag{10}$$

where $\mathcal{P}'$ in the inner minimum varies over all the 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, $(\mathcal{P}_1, \mathcal{P}_2, Y^1_{\mathsf{sp}}, Y^2_{\mathsf{sp}})$, that satisfies the following properties:

1. $\mathcal{P}_1 \sqcup \mathcal{P}_2 = \mathcal{P}$.
2. $Y_{\mathsf{sp}} \subseteq Y^1_{\mathsf{sp}} \cup Y^2_{\mathsf{sp}}$.

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 $r_P = P \cap Y_{\mathsf{sp}}$. Consider the auxiliary bipartite graph $H = (A \uplus B, E(H))$ such that the vertices in A correspond to the blocks $\{P_{11}, P_{12}, \ldots, P_{1a}, P_{21}, P_{22}, \ldots, P_{2b}\}$ and $B = Y'_1 \cup Y'_2$. An edge is added between a vertex $u \in A$ and $v \in B$, if block corresponding to u contains the intersection vertex corresponding to v. This auxiliary graph H must be a tree, in order for $(\mathcal{P}_1, \mathcal{P}_2, Y^1_{\mathsf{sp}}, Y^2_{\mathsf{sp}})$ 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:

$$J[i, D] = \min_{\substack{\mathcal{P}=\mathcal{P}_1 \sqcup \mathcal{P}_2 \\ (\mathcal{P}_1, \mathcal{P}_2, Y^1_{\mathsf{sp}}, Y^2_{\mathsf{sp}}) \in \mathbb{Q} \\ T'_1 \cup T'_2 = T'}} \left\{ \mathcal{A}[i-1, (Y, Y', \mathcal{P}_1, Y^1_{\mathsf{sp}}, T'_1)] + \mathcal{A}[i-1, (Y, Y', \mathcal{P}_2, Y^2_{\mathsf{sp}}, T'_2)] \right\} \tag{11}$$

Equation 11 corresponds to a computation in a join node.

We define $\mathcal{A}[i, D]$ for $i \in [N] \setminus \{1\}$ and type $D = (Y, Y', \mathcal{P}, Y_{\mathsf{sp}}, T')$ as,

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

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

(a) The number of connected components in F is equal to $|\mathcal{P}| = q$. Also, $V(C_i) \cap Y = P_i$.
(b) $V(F) \cap T = T'$.
(c) For each $i \in [q]$, $r_i \in P_i$.
(d) For each $i \in [q]$, and each $w \in C_i$, the r_i–w path in F is a shortest path in G and there is a shortest r–w path in G with r_i 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 S.

Lemma 14. *Let $i \in [N]$ and D be a type. Furthermore, let $A[i, D]$ be computed by the Equation 12, and have a finite value ℓ. Then there is a locally-rooted subgraph F, of type D, such that* $\mathsf{recdist}(F) \leq \ell$.

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 $D = (Y, Y', \mathcal{P}, Y_{\mathsf{sp}}, T')$ 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 $\mathsf{recdist}(F) \leq \ell$.

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 $v \notin T$, then by Equation 8, $\ell = I_v[i, D] = \mathcal{A}[i-1, (Y \setminus \{v\}, Y', \mathcal{P}', Y_{\mathsf{sp}}, T')]$. By induction hypothesis there is a locally-rooted subgraph F which is of type $D' = (Y \setminus \{v\}, Y', \mathcal{P}', Y_{\mathsf{sp}}, T')$ and $\mathsf{recdist}(F) \leq \ell$. The definition of satisfying type D or D' (conditions (a) - (d)) implies that F is of type D as well. If $v \in Y'$, then $\ell = I_v[i, D] = \mathcal{A}[i-1, D'' = (Y \setminus \{v\}, Y' \setminus \{v\}, \mathcal{P} \setminus \{v\}, Y_{\mathsf{sp}} \setminus \{v\}, T' \setminus \{v\})]$. By induction hypothesis, there is a locally-rooted subgraph F such that $\mathsf{recdist}(F) \leq \ell$ and F is of type D''. This implies that the graph $F' = F \cup (\{v\}, \emptyset)$ is of type D and $\mathsf{recdist}(F') = \mathsf{recdist}(F) \leq \ell$.

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 induction hypothesis there is a locally-rooted subgraph F which is of type D and $\mathsf{recdist}(F) \leq \ell$. On the other hand, 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 induction hypothesis we have a locally-rooted subgraph of type D and weight at most ℓ. Otherwise, $\mathcal{A}[i, D] = \mathcal{A}[i-1, (Y, Y', \mathcal{P}', Y'_{\mathsf{sp}}, T')] + \mathsf{recdist}(uv)$ for a pair $(\mathcal{P}', Y'_{\mathsf{sp}}) \in \mathbb{P}$. By induction hypothesis, there is a locally-rooted subgraph F' with type $D' = (Y, Y', \mathcal{P}', Y'_{\mathsf{sp}}, T')$, such that $\mathsf{recdist}(F') \leq \ell - \mathsf{recdist}(uv)$ 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 the properties (a)-(d). Since $\mathsf{recdist}(F) \leq \ell$ 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 the Case 1.

Case 5. Suppose $\mathcal{A}[i, D] = \mathcal{A}[i-1, (Y, Y', \mathcal{P}_1, Y^1_{\mathsf{sp}}, T'_1)] + \mathcal{A}[i-1, (Y, Y', \mathcal{P}_2, Y^2_{\mathsf{sp}}, T'_2)] = \ell$, for a tuple $(\mathcal{P}_1, \mathcal{P}_2, Y^1_{\mathsf{sp}}, Y^2_{\mathsf{sp}}) \in \mathbb{Q}$ and for $T'_1 \cup T'_2 = T'$. By induction hypothesis, for $i \in [2]$, there is a locally-rooted subgraph F_i of type $D_i = (Y, Y', \mathcal{P}_i, Y^i_{\mathsf{sp}}, T'_i)$. By the last two properties of the tuple $(\mathcal{P}_1, \mathcal{P}_2, Y^1_{\mathsf{sp}}, Y^2_{\mathsf{sp}})$, if F_1 and F_2 are forests, then $F_1 \cup F_2$ is also a forest. Also, by transitivity of the shortness property, it follows that $F = F_1 \cup F_2$ is a locally-rooted subgraph of type D. Since, $\mathsf{recdist}(F) \leq \mathsf{recdist}(F_1) + \mathsf{recdist}(F_2) \leq \ell$, 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 $c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$ 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 $\emptyset \neq P_i = X_t \cap V(C_i)$, (ii) $X_t \cap V(F) = X$, (iii) $T \cap V_t \subseteq V(F)$, and (iv) for each $i \in [q]$, and each vertex $w \in V(C_i) - \{r_i\})$, the w-r_i path in F is a shortest path in G and w has the shortness property with r_i. If there is no such subgraph F, then the value $c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$ is ∞.

Lemma 15. *Let $\mathcal{T} = (\mathbb{T}, \{X_t\}_{t \in V(\mathbb{T})})$ be a nice tree decomposition of S. 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 of $\mathcal{P}$. Let V_t be the union of bags in the subtree rooted at t, and $T' = T \cap V_t$. Then $A[i, (X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')] \leq c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$.*

Proof. Consider a nice tree decomposition $\mathcal{T} = (\mathbb{T}, \{X'_t\}_{t \in V(\mathbb{T})})$, of S, rooted at a node z. 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 S. Note that all the leaf bags and the root bag z, 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 z, have level 1. The level of z is the height of $\mathbb{T}$. 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 $(V_t, \{e \mid e \text{ is introduced in a the subtree rooted at } t\})$ 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 $X_{\mathsf{sp}} = \{r_1, \ldots, r_q\}$ such that $r_i \in P_i$, $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T \cap V_t)] \leq c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$. 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 $\mathcal{A}[1, (X_t, X, \{X\}, X, T \cap V_t)] = 0 = c[t, X, \{X\}, X]$. Otherwise $\mathcal{A}[1, (X_t, X, \mathcal{P}, X, T \cap V_t)] = \infty = c[t, X, \{X\}, X]$. 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 $T' = T \cap V_t$. If $(X_t \setminus X) \cap T \neq \emptyset$, then by definition $c[t, X, \mathcal{P}, X_{\mathsf{sp}}] = \infty$ and so the statement holds. Suppose $(X_t \setminus X) \cap T = \emptyset$. Since $\widehat{S}$ is a Steiner arborescence for T, $T \subseteq V(\widehat{S})$. Since X_t is a bag in the tree decomposition $\mathcal{T}$, each terminal in a connected component C of $\widehat{S} - X_t$ 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 $C_1, \ldots, C_j$ of $\widehat{S} - X_t$ such that $T' = T \cap V_t = T \cap (X_t \cup \bigcup_{i=1}^{j} V(C_i))$. This implies that $(X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')$ is a type. Let $\mathcal{P} = \{P_1, \ldots, P_q\}$, $X_{\mathsf{sp}} = \{r_1, \ldots, r_q\}$ such that $r_i \in P_i$, and F be a witness subgraph for the value $c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$. That is, $\mathsf{recdist}(F) = c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$ and the following conditions holds: (i) F has q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_i = X_t \cap V(C_i)$, (ii) $X_t \cap V(F) = X$, (iii) $T \cap V_t \subseteq V(F)$, and (iv) for each $i \in [q]$, and each vertex $w \in V(C_i) - \{r_i\})$, the w-r_i path in F is a shortest path in G and w has the shortness property with r_i. 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 $c[t', X, \mathcal{P}, X_{\mathsf{sp}}] \leq \mathsf{recdist}(F)$. By Equations 8 and 12, $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')]$. By induction hypothesis,

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

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 $c[t', X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, X_{\mathsf{sp}} \setminus \{v\}] \leq \mathsf{recdist}(F)$. By Equations 8 and 12,

$$\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, X_{\mathsf{sp}} \setminus \{v\}, T')].$$

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

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

$$\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')] \leq \mathcal{A}[\ell_t - 1, (X'_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')].$$

By induction hypothesis,

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

Suppose $uv \in E(F)$. This means that there is a single component C, in F, that contains u, v. Let $r_C = C \cap X_{\mathsf{sp}}$. 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 $X'_{\mathsf{sp}} = X_{\mathsf{sp}} \cup \{v\}$. This implies that $c[t', X, \mathcal{P}', X'_{\mathsf{sp}}] \leq \mathsf{recdist}(F \setminus \{uv\}) = \mathsf{recdist}(F) - \mathsf{recdist}(uv)$. By induction hypothesis,

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

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 $(\mathcal{P}', X'_{\mathsf{sp}}) \in \mathbb{P}$. Thus, by Equations 9 and 12,

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

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 $c[t', X, \mathcal{P}, X_{\mathsf{sp}}] \leq \mathsf{recdist}(F)$. By induction hypothesis, $\mathcal{A}[\ell_t, (X_{t'}, X, \mathcal{P}, X_{\mathsf{sp}}, T)] \leq c[t', X, \mathcal{P}, X_{\mathsf{sp}}] \leq \mathsf{recdist}(F)$. By Equations 10 and 12, $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')] \leq \mathcal{A}[\ell_t - 1, (X_{t'}, X, \mathcal{P}, X_{\mathsf{sp}}, T')] \leq \mathsf{recdist}(F)$.

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

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

By Equations 10 and 12,

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

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 $X_{t_i}, i \in \{1, 2\}$, is $\ell_t - 1$. 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 the connected components in $V(F)$ contain at least one vertex from X and X_t is a bag in the tree decomposition, all the connected components in F_1 as well as in F_2 contains 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 $\mathcal{P}_2 = \{X \cap V(C''_i), \ldots, X \cap V(C''_{q''})\}$. 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 $r_C \in C \cap X_{\mathsf{sp}}$. Then, each vertex in $V(C)$ has the shortness property with $r_{C'}$. This way, we obtain, for each $i \in [2]$, a set X^i_{sp} defined by taking exactly one vertex from each block of $\mathcal{P}_i$. Thus, $c[t_1, X, \mathcal{P}_1, X^1_{\mathsf{sp}}] \leq \mathsf{recdist}(F_1)$ and $c[t_2, X, \mathcal{P}_2, X^2_{\mathsf{sp}}] \leq \mathsf{recdist}(F_2)$. By induction hypothesis, $\mathcal{A}[\ell_{t_i}, (X_{t_i}, X, \mathcal{P}_i, X^i_{\mathsf{sp}}, T'_i)] \leq c[t_i, X, \mathcal{P}_i, X^i_{\mathsf{sp}}]$ for $i \in \{1, 2\}$. The definitions of F, F_1 and F_2 implies that $\mathcal{P} = \mathcal{P}_1 \sqcup \mathcal{P}_2$. Also, notice that the tuple $(\mathcal{P}_1, \mathcal{P}_2, X^1_{\mathsf{sp}}, X^2_{\mathsf{sp}}) \in \mathbb{Q}$. By Equations 11 and 12, $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')] \leq \mathcal{A}[\ell_t - 1, (X_t, X, \mathcal{P}_1, X^1_{\mathsf{sp}}, T'_1)] + \mathcal{A}[\ell_t - 1, (X_t, X, \mathcal{P}_2, X^2_{\mathsf{sp}}, T'_2)] \leq \mathsf{recdist}(F_1) + \mathsf{recdist}(F_2) = \mathsf{recdist}(F)$. This concludes the proof. □

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

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

Proof. We take as input a set T of n terminal points, the Hanan grid G of T and the weight function $\mathsf{recdist}$. Furthermore, $r \in T$is the root terminal. Then using Lemma 9 we compute a shortest path RSA $\widehat{S}$. By Lemma 12, we know that there is an optimal Steiner tree

S_{opt} with $\mathsf{tw}(\widehat{S} \cup S_{\mathsf{opt}}) \leq 36\sqrt{n}$ Based on the shortest path RSA $\widehat{S}$, we apply Lemma 13, 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 $\min_{i\in[N]} \mathcal{A}[i, (\{r\}, \{r\}, \{\{r\}\}, \{r\}, T)]$.

For the hypothetical set S, fix an optimal nice tree decomposition $\mathcal{T}$, rooted at node z. Add the root terminal r to each bag of the nice tree decomposition (As described in the proof of Lemma 15). The treewidth of this tree decomposition is bounded by $36\sqrt{n} + 1$. 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 $T' = T \cap V_t$. 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, $c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$ is the size of a locally-rooted subgraph of type $(X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')$. Then, Lemmata 14 and 15 imply that $\mathcal{A}[\ell_t, (X_t, X, \mathcal{P}, X_{\mathsf{sp}}, T')] = c[t, X, \mathcal{P}, X_{\mathsf{sp}}]$. In particular, for the root z of the tree decomposition, $\mathcal{A}[\ell_z, (\{r\}, \{r\}, \{\{r\}\}, \{r\}, T)] = c[z, \{r\}, \{\{r\}\}, \{r\}]$. Notice, that $c[z, \{r\}, \{\{r\}\}, \{r\}]$ is the size of a minimum rectilinear Steiner arborescence of G.

On the other hand, by Lemma 14, for all $i \in [N]$, if $A[i, (\{r\}, \{r\}, \{\{r\}\}, \{r\}, T)] = \ell$ then there is a locally-rooted subgraph F that connects all the terminals of T, and that satisfies $\mathsf{recdist}(F) \leq \ell$. As the algorithm outputs $\min_{i\in[N]} \mathcal{A}[i, (\{r\}, \{r\}, \{\{r\}\}, \{r\}, T)]$, 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 $N \cdot 2^{\mathcal{O}(\sqrt{n}\log n)}$ and each entry can be filled in time $2^{\mathcal{O}(\sqrt{n}\log n)} n^{\mathcal{O}(1)}$. Thus, the running time of the algorithm is $2^{\mathcal{O}(\sqrt{n}\log n)} n^{\mathcal{O}(1)}$. Using standard back-tracking tricks we can also output an optimal RSA. This concludes the proof. □

5 Conclusion

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

References

[1] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, *Fourier meets Möbius: fast subset convolution*, in Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC), New York, 2007, ACM, pp. 67–74. 2

[2] M. Brazil and M. Zachariasen, *Optimal Interconnection Trees in the Plane: Theory, Algorithms and Applications*, Springer, 2015. 1, 2

[3] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer-Verlag, Berlin, 2015. 3, 5, 15, 16, 17

[4] L. L. Deneen, G. M. Shute, and C. D. Thomborson, *A probably fast, provably optimal algorithm for rectilinear Steiner trees*, Random Structures Algorithms, 5 (1994), pp. 535–557. 2

[5] S. E. Dreyfus and R. A. Wagner, *The Steiner problem in graphs*, Networks, 1 (1971), pp. 195–207. 2

[6] B. Fuchs, W. Kern, D. Mölle, S. Richter, P. Rossmanith, and X. Wang, *Dynamic programming for minimum Steiner trees*, Theory of Computing Systems, 41 (2007), pp. 493–500. 2

[7] J. L. Ganley, *Computing optimal rectilinear Steiner trees: a survey and experimental evaluation*, Discrete Appl. Math., 90 (1999), pp. 161–171. 2

[8] J. L. Ganley and J. P. Cohoon, *Improved computation of optimal rectilinear Steiner minimal trees*, Internat. J. Comput. Geom. Appl., 7 (1997), pp. 457–472. 2

[9] M. R. Garey and D. S. Johnson, *The rectilinear Steiner tree problem is NP-complete*, SIAM J. Appl. Math., 32 (1977), pp. 826–834. 2

[10] Q.-P. Gu and H. Tamaki, *Improved bounds on the planar branchwidth with respect to the largest grid minor size*, Algorithmica, 64 (2012), pp. 416–453. 8

[11] M. Hanan, *On Steiner's problem with rectilinear distance*, SIAM J. Appl. Math., 14 (1966), pp. 255–265. 1, 2

[12] F. K. Hwang, *On Steiner minimal trees with rectilinear distance*, SIAM J. Appl. Math., 30 (1976), pp. 104–114. 3

[13] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner tree problem*, vol. 53 of Annals of Discrete Mathematics, North-Holland Publishing Co., Amsterdam, 1992. 1, 2

[14] P. N. Klein and D. Marx, *A subexponential parameterized algorithm for Subset TSP on planar graphs*, in Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2014, pp. 1812–1830. 3

[15] L. Nastansky, S. M. Selkow, and N. F. Stewart, *Cost-minimal trees in directed acyclic graphs*, Z. Operations Res. Ser. A-B, 18 (1974), pp. A59–A67. 3

[16] J. Nederlof, *Fast polynomial-space algorithms using inclusion-exclusion*, Algorithmica, 65 (2013), pp. 868–884. 2

[17] M. Pilipczuk, M. Pilipczuk, P. Sankowski, and E. J. van Leeuwen, *Subexponential-time parameterized algorithm for Steiner tree on planar graphs*, in Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS), vol. 20 of Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2013, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 353–364. 2

[18] M. Pilipczuk, M. Pilipczuk, P. Sankowski, and E. J. van Leeuwen, *Network sparsification for Steiner problems on planar and bounded-genus graphs*, in Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2014, pp. 276–285. 2

[19] H. J. Prömel and A. Steger, *The Steiner Tree Problem*, Advanced Lectures in Mathematics, Friedr. Vieweg & Sohn, Braunschweig, 2002. 1

[20] N. Robertson, P. D. Seymour, and R. Thomas, *Quickly excluding a planar graph*, J. Combinatorial Theory Ser. B, 62 (1994), pp. 323–348. 8

[21] W. Shi and C. Su, *The rectilinear steiner arborescence problem is np-complete*, SIAM J. Comput., 35 (2005), pp. 729–740. 3

[22] C. D. Thomborson, L. L. Deneen, and G. M. Shute, *Computing a rectilinear steiner minimal tree in $n^{O(\sqrt{n})}$ time*, in Parallel Algorithms and Architectures, Springer, 1987, pp. 176–183. 2