A Fast Algorithm for Computing a Planar Support for Non-Piercing Rectangles

Ambar Pal
Johns Hopkins University, Baltime, MD, USA

Rajiv Raman
Indraprastha Institute of Information Technology Delhi, New Delhi, India

Saurabh Ray
NYU Abu Dhabi, UAE

Karamjeet Singh
Indraprastha Institute of Information Technology Delhi, New Delhi, India

Abstract

For a hypergraph $\mathcal{H} = (X, \mathcal{E})$ a *support* is a graph G on X such that for each $E \in \mathcal{E}$, the induced subgraph of G on the elements in E is connected. If G is planar, we call it a planar support. A set of axis parallel rectangles $\mathcal{R}$ forms a non-piercing family if for any $R_1, R_2 \in \mathcal{R}$, $R_1 \setminus R_2$ is connected.

Given a set P of n points in $\mathbb{R}^2$ and a set $\mathcal{R}$ of m *non-piercing* axis-aligned rectangles, we give an algorithm for computing a planar support for the hypergraph $(P, \mathcal{R})$ in $O(n \log^2 n + (n+m) \log m)$ time, where each $R \in \mathcal{R}$ defines a hyperedge consisting of all points of P contained in R.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Algorithms, Hypergraphs, Computational Geometry, Visualization

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.53

1 Introduction

Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$, a *support* is a graph G on V such that for all $E \in \mathcal{E}$, the subgraph induced by E in G, denoted $G[E]$ is connected. The notion of a support was introduced by Voloshina and Feinberg [30] in the context of VLSI circuits. Since then, this notion has found wide applicability in several areas, such as visualizing hypergraphs [6, 7, 8, 9, 11, 19, 21], in the design of networks [1, 3, 4, 13, 20, 23, 26], and similar notions have been used in the analysis of local search algorithms for geometric problems [2, 5, 15, 24, 25, 27].

Any hypergraph clearly has a support: the complete graph on all vertices. In most applications however, we require a support with an additional structure. For example, we may want a support with the fewest number of edges, or a support that comes from a restricted family of graphs (e.g., outerplanar graphs).

Indeed, the problem of constructing a support has been studied by several research communities. For example, Du, et al., [16, 17, 18] studied the problem of minimizing the number of edges in a support, motivated by questions in the design of vacuum systems. The problem has also been studied under the topic of "minimum overlay networks" [20, 14] with applications to distributed computing. Johnson and Pollack [22] showed that it is NP-hard to decide if a hypergraph admits a planar support.

© Ambar Pal, Rajiv Raman, Saurabh Ray, and Karamjeet Singh;
licensed under Creative Commons License CC-BY 4.0

In another line of work, motivated by the analysis of approximation algorithms for packing and covering problems on *geometric hypergraphs*[1], several authors have considered the problem of constructing supports that belong to a family having sublinear sized separators[2] such as planar graphs, or graphs of bounded genus [27, 28, 29]. For this class of problems, the problem of interest is only to show the existence of a support from a restricted family of graphs.

Our contribution. So far there are very few tools or techniques to construct a support for a given hypergraph or even to show that a support with desired properties (e.g., planarity) exists. Our paper presents a fast algorithm to construct a planar support for a restricted setting, namely hypergraphs defined by axis-parallel rectangles that are *non-piercing*, i.e., for each pair of intersecting rectangles, one of them contains a corner of the other. This may seem rather restrictive. However, even if we allow each rectangle to belong to at most one piercing pair of rectangles, it is not difficult to construct examples where for any $r \geq 3$, any support must have $K_{r,r}$ as a topological minor. To see this, consider a geometric drawing of $K_{r,r}$ in the usual manner, i.e., the two partite sets on two vertical lines, and the edges as straight-line segments. Replace each edge of the graph by a long path, and then replace each edge along each path by a small rectangle that contains exactly two points. Where the edges cross, a pair of rectangles corresponding to each edge cross. Since each rectangle contains two points, it leaves us no choice as to the edges we can add. It is easy to see that the resulting support contains $K_{r,r}$ as a topological minor. Further, even for this restricted problem, the analysis of our algorithm is highly non-trivial, and we hope that the tools introduced in this paper will be of wider interest.

Raman and Ray [27], showed that the hypergraph defined by *non-piercing regions*[3] in the plane admits a planar support. Their proof implies an $O(m^2(\min\{m^3, mn\} + n))$ time algorithm to compute a planar support where m is the number of regions and n is the number of points in the arrangement of the regions. While their algorithm produces a plane embedding, the edges may in general be arbitrarily complicated curves i.e., they may have an arbitrary number of bends. It can be shown that if the non-piercing regions are convex then there exists an embedding of the planar support with straight edges but it is not clear how to find such an embedding efficiently.

We present a simple and fast algorithm for drawing plane supports with straight-line edges for non-piercing rectangles. More precisely, the following is the problem definition:

Support for non-piercing rectangles:
Input: A set of m axis-parallel non-piercing rectangles $\mathcal{R}$ and a set P of n points in $\mathbb{R}^2$.
Output: A plane graph G on P s.t. for each $R \in \mathcal{R}$, $G[R \cap P]$, namely the induced subgraph on the points in $R \cap P$, is connected.

Our algorithm runs in $O(n \log^2 n + (n + m) \log m)$ time, and can be easily implemented using existing data structures. The embedding computed by our algorithm not only has straight-line edges but also for each edge e, the axis-parallel rectangle with e as the diagonal does not contain any other point of P – this makes the visualization cleaner.

[1] In a geometric hypergraph, the elements of the hypergraph are points in the plane, and the hyperedges are defined by geometric regions in the plane, where each region defines a hyperedge consisting of all points contained in the region.

[2] A family of graphs $\mathcal{G}$ admits sublinear sized separators if there exist $0 < \alpha, \beta < 1$ s.t. for any $G \in \mathcal{G}$, there exists a set $S \subseteq V(G)$ s.t. $G[V \setminus S]$ consists of two parts A and B with $|A|, |B| \leq \alpha|V|$, and there is no path in $G[V \setminus S]$ between a vertex in A to a vertex in B. Further, $|S| \leq |V|^{1-\beta}$.

[3] A family of simply connected regions $\mathcal{R}$, each of whose boundary is defined by a simple Jordan curve is called non-piercing if for every pair of regions $A, B \in \mathcal{R}$, $A \setminus B$ and $B \setminus A$ are connected. The result of [27] was for more general families.

In order to develop a faster algorithm, we need to find a new construction (different from [27]), and the proof of correctness for this construction is not so straightforward. We use a sweep line algorithm. However, at any point in time, it is not possible to have the invariant that the current graph is a support for the portions of the rectangles that lie to the left of the sweep line. Instead, we show that certain *slabs* within each rectangle induce connected components of the graph and only after we sweep over a rectangle completely do we finally have the property that the set of points in that rectangle induce a connected subgraph.

Organization. The rest of the paper is organized as follows. We start in Section 2 with related work. In Section 3, we present preliminary notions required for our algorithm. In Section 4, we present a fast algorithm to construct a planar support. We show in Section 4.1 that the algorithm is correct, i.e., it does compute a planar support. We present the implementation details in Section 4.2.

2 Related work

The notion of the existence of a support, and in particular a planar support arose in the field of VLSI design [30]. A VLSI circuit is viewed as a hypergraph where each individual electric component corresponds uniquely to a vertex of the hypergraph, and sets of components called *nets* correspond uniquely to a hyperedge. The problem is to connect the components with wires so that for every net, there is a tree spanning its components. Note that planarity in this context is natural as we don't want wires to cross.

Thus, a motivation to study supports was to define a notion of planarity suitable for hypergraphs. Unlike for graphs, there are different notions of planarity of hypergraphs, not all equivalent to each other. Zykov [32] defined a notion of planarity that was more restricted. A hypergraph is said to be Zykov-planar if its incidence bipartite graph is planar [32, 31].

Johnson and Pollack [22] showed that deciding if a hypergraph admits a planar support is NP-hard. The NP-hardness result was sharpened by Buchin, et al., [11] who showed that deciding if a hypergraph admits a support that is a k-outerplanar graph, for $k \geq 2$ is NP-hard, and showed that we can decide in polynomial time if a hypergraph admits a support that is a tree of bounded degree. Brandes, et al., [8] showed that we can decide in polynomial time if a hypergraph admits a support that is a cactus[4].

Brandes et al., [9], motivated by the drawing of metro maps, considered the problem of constructing *path-based supports*, which must satisfy an additional property that the induced subgraph on each hyperedge contains a Hamiltonian path on the vertices of the hyperedge.

Another line of work, motivated by the analysis of approximation algorithms for packing and covering problems on geometric hypergraphs started with the work of Chan and Har-Peled [12], and Mustafa and Ray [25]. The authors showed, respectively, that for the Maximum Packing[5] of pseudodisks[6], and for the Hitting Set[7] problem for pseudodisks, a simple *local search* algorithm yields a PTAS. These results were extended by Basu Roy, et

[4] A cactus is a graph where each edge of the graph lies in at most one cycle.
[5] In a Maximum Packing problem, the goal is to select the largest subset of pairwise disjoint hyperedges of a hypergraph.
[6] A set of simple Jordan curves is a set of pseudocircles if the curves pairwise intersect twice or zero times. The pseudocircles along with the bounded region defined by the curves is a collection of pseudodisks.
[7] In the Hitting Set problem, the goal is to select the smallest subset of vertices of a hypergraph so that each hyperedge contains at least one vertex in the chosen subset.

al., [5] to work for the Set Cover and Dominating Set[8] problems defined by points and non-piercing regions, and by Raman and Ray [27], who gave a general theorem on the existence of a planar support for any geometric hypergraph defined by two families of *non-piercing regions*. This result generalized and unified the previously mentioned results, and for a set of m non-piercing regions, and a set of n points in the plane, it implies that a support graph can be constructed in time $O(m^2(\min\{m^3, mn\} + n))$. It follows that for non-piercing axis-parallel rectangles, a planar support can be constructed in time $O(m^2(\min\{m^3, mn\} + n))$. However, in the embedding of the support thus constructed, the edges may be drawn as arbitrary curves.

3 Preliminaries

Let $\mathcal{R} = \{R_1, \ldots, R_m\}$ denote a set of axis-parallel rectangles and let $P = \{p_1, \ldots, p_n\}$ denote a set of points in the plane. We assume that the rectangles and points are in *general position*, i.e., the points in P have distinct x and y coordinates, and the boundaries of any two rectangles in $\mathcal{R}$ are defined by distinct x-coordinates and distinct y-coordinates. Further, we assume that no point in P lies on the boundary of a rectangle in $\mathcal{R}$.

Piercing, Discrete Piercing. A rectangle R' is said to *pierce* a rectangle R if $R \setminus R'$ consists of two connected components. A collection $\mathcal{R}$ of rectangles is *non-piercing* if no pair of rectangles pierce. A rectangle R' *discretely pierces* a rectangle R if R' pierces R and each component of $R \setminus R'$ contains a point of P. Since we are primarily concerned with discrete piercing, the phrase "R pierces R'" will henceforth mean discrete piercing, unless stated otherwise. Note that while piercing is a symmetric relation, discrete piercing is not.

"L"-shaped edge. We construct a drawing of a support graph G on P using "L"-shaped edges of type: ⌉ or ⌋. Henceforth, the term *edge* will mean one of the two "L"-shaped edges joining two points. The embedded graph may not be planar due to the overlap of the edges along their horizontal/vertices segments. However, as we show, G satisfies the additional property that for each edge, the axis-parallel rectangle defined by the edge has no points of P in its interior (formal definition below), and that no pair of edges cross. Consequently, replacing each edge with the straight segment joining its end-points yields a plane embedding of G.

Delaunay edge, Valid edge, $\mathbf{R(\cdot), h(\cdot), v(\cdot)}$. For an edge between points $p, q \in P$, let $R(pq)$ denote the rectangle with diagonally opposite corners p and q. The edge pq is a *Delaunay edge* if the interior of $R(pq)$ does not contain a point of P. We say that an edge pq (discretely) pierces a rectangle R if $R \setminus \{pq\}$ consists of two regions, and each region contains a point of P. An edge pq is said to be *valid* if it does not discretely pierce any rectangle $R \in \mathcal{R}$, and does not cross any existing edge. For an edge pq, we use $h(pq)$ for the horizontal segment of pq, and $v(pq)$ for the vertical segment of pq.

Monotone Path, Point above Path. A path π is said to be x-monotone if a vertical line, i.e., a line parallel to the y-axis, does not intersect the path in more than one point. We modify this definition slightly for our purposes – we say that a path consisting of a sequence

[8] In the Set Cover problem, the input is a set system $(X, \mathcal{S})$ and the goal is to select the smallest sub-collection $\mathcal{S}'$ that covers the elements in X. For a graph, a subset of vertices S is a dominating set if each vertex in the graph is either in S, or is adjacent to a vertex in S.

of $\urcorner$, or $\lrcorner$ edges is x-monotone if any vertical line intersects the path in at most one vertical segment (which may in some cases be a single point). Let π be a path and q be a point not on the path. We say that "q lies above π" if ℓ_q, the vertical line through q intersects π at point(s) below q. We define the notion that "q lies below π" analogously. Note that these notions are defined only if ℓ_q intersects π.

Left(Right)-Neighbor, Left(Right)-Adjacent. For a point $q \in P$ and a set $P' \subseteq P$, the *right-neighbor* of q in P' is q_1, where $q_1 = \text{argmin}_{q' \in P'}\{x(q') : x(q') > x(q)\}$. The *left-neighbor* of q in P' is defined similarly, i.e., q_0 is the left-neighbor of q, where $q_0 = \text{argmax}_{q' \in P'}\{x(q') : x(q') < x(q)\}$. Note that being a left- or right-neighbor is a *geometric notion*, and not related to the support graph we construct. We use the term *left-adjacent* to refer to the neighbors of q in a plane graph G that lie to the left of q. The term *right-adjacent* is defined analogously.

4 Algorithm

In this section, we present an algorithm to compute a planar support for the hypergraph defined by points and non-piercing axis-parallel rectangles in $\mathbb{R}^2$: perform a left-to-right vertical line sweep and at each input point encountered, add all possible *valid Delaunay edges* to previous points. The algorithm, presented as Algorithm 1, draws edges having shapes in $\{\urcorner, \lrcorner\}$. We prove correctness of Algorithm 1 in Section 4.1, and show how it can be implemented to run in $O(n \log^2 n + (n+m)\log m))$ time in Section 4.2.

Algorithm 1 The algorithm outputs a graph G on P embedded in $\mathbb{R}^2$, whose edges are valid Delaunay edges of type $\{\urcorner, \lrcorner\}$. Replacing each Delaunay edge $\{p, q\}$ by the diagonal of $R(pq)$ yields a plane embedding of G.

Input: A set P of points, and a set $\mathcal{R}$ of non-piercing axis-parallel rectangles in $\mathbb{R}^2$.
Output: Embedded Planar Support $G = (P, E)$
Order P in increasing order of *x-coordinates*: $(p_1, \ldots, p_n)$
$E = \emptyset$
for each point p_i in sorted order, $i \in \{2, 3, \ldots, n\}$ **do**
  $E = E \cup \{e_{ij} = p_i p_j \mid j < i,$ and e_{ij} is a *valid Delaunay edge.*$\}$
end

4.1 Correctness

In this section, we show that the graph G constructed on P by Algorithm 1 is a support graph for the rectangles in $\mathcal{R}$, and this is sufficient as planarity follows directly by construction. The proof is technical, and we start with some necessary notation.

For a rectangle R, we denote the y-coordinates of the lower and upper horizontal sides by $y_-(R)$ and $y_+(R)$, respectively. Similarly, $x_-(R)$ and $x_+(R)$ denote respectively the x-coordinates of the left and right vertical sides. We denote the vertical line through any point p by ℓ_p.

We use $\textsc{Piece}(R, H)$ to denote the rectangle $R \cap H$ for a halfplane H defined by a vertical line. We abuse notation and use $\textsc{Piece}(R, p)$ to denote the rectangle $R \cap H_-(\ell_p)$, the intersection of R with the left half-space defined by the vertical line through the point p.

We also use the notation $R[x_-, x_+]$ to denote the sub-rectangle of rectangle R, that lies between x-coordinates x_- and x_+. Similarly, we use $R[y_-, y_+]$ to denote the sub-rectangle of R that lies between the y-coordinates y_- and y_+.

To avoid boundary conditions in the definitions that follow, we add two rectangles: R_{top} above all rectangles in $\mathcal{R}$, and R_{bot} below all rectangles in $\mathcal{R}$, that is $y_-(R_{top}) > \max_{R\in\mathcal{R}} y_+(R)$, and $y_+(R_{bot}) < \min_{R\in\mathcal{R}} y_-(R)$. The rectangles R_{top}, and R_{bot} span the width of all rectangles, i.e., $x_-(R_{top}) = x_-(R_{bot}) < \min_{R\in\mathcal{R}} x_-(R)$, and $x_+(R_{top}) = x_+(R_{bot}) > \max_{R\in\mathcal{R}} x_+(R)$. We add two points $P_{top} = \{p_1^+, p_2^+\}$ to the interior of R_{top}, and two points $P_{bot} = \{p_1^-, p_2^-\}$ to the interior of R_{bot}, such that $x(p_1^+) = x(p_1^-) < \min_{p\in P} x(p)$, and $x(p_2^+) = x(p_2^-) > \max_{p\in P} x(p)$. Let $\mathcal{R}' = \mathcal{R} \cup \{R_{top}, R_{bot}\}$, and $P' = P \cup P_{top} \cup P_{bot}$. For ease of notation, we simply use $\mathcal{R}$ and P to denote $\mathcal{R}'$ and P' respectively, and implicitly assume the existence of $R_{top}, R_{bot}, P_{top}$ and P_{bot}.

For a vertical segment s, a rectangle $R \in \mathcal{R}'$ is said to be *active* at s if it is either discretely pierced by s i.e., $R \setminus s$ is not connected and each of the two components contains a point of P, or there is a point of $P \cap s$ in R. We denote the set of all active rectangles at s by $\textsc{Active}(s)$. For a point $p \in P \cap s$, we define $\textsc{Contain}(s,p)$ to be the set of rectangles in $\textsc{Active}(s)$ that contains the point p. We define $\textsc{Above}(s,p)$ to be the set of rectangles in $\textsc{Active}(s)$ that lie strictly above p, i.e., $\textsc{Above}(s,p) = \{R \in \textsc{Active}(s) : y_-(R) > y(p)\}$. Similarly, $\textsc{Below}(s,p) = \{R \in \textsc{Active}(s) : y_+(R) < y(p)\}$. It follows that for any point $p \in s$, $\textsc{Active}(s) = \textsc{Contain}(s,p) \sqcup \textsc{Above}(s,p) \sqcup \textsc{Below}(s,p)$, where $\sqcup$ denotes disjoint union.

Note that for the vertical line ℓ_p through $p \in P$, $\textsc{Active}(\ell_p) \neq \emptyset$, as $\textsc{Active}(\ell_p)$ contains the rectangles R_{top} and R_{bot}. Similarly, $\textsc{Above}(\ell_p, p) \neq \emptyset$ and $\textsc{Below}(\ell_p, p) \neq \emptyset$. Abusing notations slightly, we write $\textsc{Active}(p)$ instead of $\textsc{Active}(\ell_p)$, and likewise with $\textsc{Contain}(\cdot), \textsc{Above}(\cdot)$ and $\textsc{Below}(\cdot)$.

For a point $p \in P$, we now introduce the notion of *barriers*. Any active rectangle R' in $\textsc{Above}(p)$ prevents a valid Delaunay edge incident on p from being incident to a point to the left of p above $y_+(R')$, as such an edge would discretely pierce R'. Hence, among all rectangles $R' \in \textsc{Above}(p)$, the one with lowest $y_+(R')$ is called the *upper barrier* at p, denoted $\textsc{Ub}(p)$. Thus, $\textsc{Ub}(p) = \operatorname{argmin}_{R'\in\textsc{Above}(p)} y_+(R')$.

Similarly, we define the *lower barrier* of p, $\textsc{Lb}(p) = \operatorname{argmax}_{R'\in\textsc{Below}(p)} y_-(R')$.

Note that $\textsc{Ub}(p)$ and $\textsc{Lb}(p)$ exist for any $p \in P$ since $\textsc{Above}(p)$ and $\textsc{Below}(p)$ are non-empty.

While the rectangles in $\mathcal{R}'$ are non-piercing, a rectangle $R' \in \textsc{Active}(p)$ can be discretely pierced by $\textsc{Piece}(R,p)$. We thus define the *upper piercing barrier* $\textsc{Upb}(R,p)$ as the rectangle $R' \in \textsc{Above}(p)$ with the lowest $y_+(R')$ that is pierced by $\textsc{Piece}(R,p)$, and we define the *lower piercing barrier* $\textsc{Lpb}(R,p)$ analogously. That is,

$$\textsc{Upb}(R,p) = \operatorname*{argmin}_{\substack{R'\in\textsc{Above}(p)\\ \textsc{Piece}(R,p)\text{ pierces }R'}} y_+(R') \quad\text{and}\quad \textsc{Lpb}(R,p) = \operatorname*{argmax}_{\substack{R'\in\textsc{Below}(p)\\ \textsc{Piece}(R,p)\text{ pierces }R'}} y_-(R')$$

For a point $p \in P$ and a rectangle $R \in \textsc{Contain}(p)$, if $\textsc{Upb}(R,p)$ or $\textsc{Lpb}(R,p)$ exist, then the horizontal line containing $y_+(\textsc{Upb}(R,p))$ together with the horizontal line containing $y_-(\textsc{Lpb}(R,p))$ naturally split $\textsc{Piece}(R,p)$ into at most three sub-rectangles called *slabs*. The point p lies in exactly one of these slabs, denoted $\textsc{Slab}(R,p)$. Thus, $\textsc{Slab}(R,p)$ is the sub-rectangle of R whose left and right-vertical sides are respectively defined by $x_-(R)$ and ℓ_p, and the upper and lower sides are respectively defined by

$$y_+(\textsc{Slab}(R,p)) = \begin{cases} y_+(\textsc{Upb}(R,p)), & \text{if } \textsc{Upb}(R,p) \text{ exists}\\ y_+(R), & \text{otherwise}\end{cases}$$

and similarly,

$$y_-(\textsc{Slab}(R,p)) = \begin{cases} y_-(\textsc{Lpb}(R,p)), & \text{if } \textsc{Lpb}(R,p) \text{ exists}\\ y_-(R), & \text{otherwise}\end{cases}$$

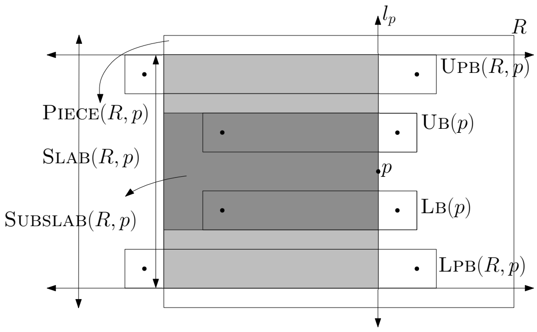


Figure 1 The figure above shows $\text{UB}(p), \text{LB}(p)$, and the upper and lower piercing barriers $\text{LPB}(R,p)$ and $\text{UPB}(R,p)$ of $\text{PIECE}(R,p)$. The slab $\text{SLAB}(R,p)$ containing p defined by $\text{UPB}(R,p)$ and $\text{LPB}(R,p)$ is shaded. The dark grey part shows the $\text{SUBSLAB}(R,p)$.

By definition, for a point p and $R \in \text{CONTAIN}(p)$, if $\text{UPB}(R,p)$ exists, then $y_+(\text{UPB}(R,p)) \geq y_+(\text{UB}(p))$. Similarly, if $\text{LPB}(R,p)$ exists, then $y_-(\text{LPB}(R,p)) \leq y_-(\text{LB}(p))$. Thus, $y_+(\text{UB}(p))$ and $y_-(\text{LB}(p))$ together split $\text{SLAB}(R,p)$ further into at most 3 sub-rectangles called *sub-slabs* whose vertical sides coincide with the vertical sides of $\text{SLAB}(R,p)$, and the horizontal sides are defined by $y_+(\text{UB}(p))$ and $y_-(\text{LB}(p))$. Let $\text{SUBSLAB}(R,p)$ denote the sub-slab containing p. Figure 1 illustrates the notions defined thus far. Note that the left-adjacent vertices of p in G that are contained in R, only lie in $\text{SUBSLAB}(R,p)$.

Proof Strategy. To prove that the graph G constructed by Algorithm 1 is a support for $\mathcal{R}$, we proceed in two steps. First (and the part that requires most of the work) we show that for each $R \in \mathcal{R}$ and $p \in P \cap R$, the subgraph of G induced by the points in $\text{SLAB}(R,p)$ is connected. Second, we show that if p is the rightmost point in R, then $\text{SLAB}(R,p)$ contains all points in $R \cap P$ which, by the first part, is connected.

When processing a point p, Algorithm 1 only adds valid Delaunay edges from p to points to its left. That is, we only add edges to a subset of points in $\text{SUBSLAB}(R,p)$. To show that $\text{SLAB}(R,p)$ is connected, one approach could be to show that the $\text{SLAB}(R,p)$ is covered by sub-slabs defined by points in $\text{SLAB}(R,p)$, adjacent sub-slabs share a point of P, and that points in a sub-slab induce a connected subgraph. Unfortunately, this is not true, and we require a finer partition of a slab. We proceed as follows: First, we define a sequence of sub-rectangles of $\text{SLAB}(R,p)$ called *strips*, denoted $\text{STRIP}(R,p,i)$ for $i \in \{-t,\ldots,k\}$, where the strips that lie above p have positive indices, the strips that lie below p have negative indices, and the unique strip that contains p has index 0. Further, each strip shares its vertical sides with $\text{SLAB}(R,p)$. In the following, since R and p are fixed, we refer to $\text{STRIP}(R,p,i)$ as STRIP_i. We define the strips so that they satisfy the following conditions:

- **(s-i)** Each strip is contained in the slab, i.e, $\text{STRIP}_i \subseteq \text{SLAB}(R,p)$ for each $i \in \{-t,\ldots,k\}$.
- **(s-ii)** The union of strips cover the slab, i.e., $\text{SLAB}(R,p) \subseteq \cup_{i=-t}^{k}\text{STRIP}_i$, and
- **(s-iii)** Consecutive strips contain a point of P in their intersection, i.e., $\text{STRIP}_i \cap \text{STRIP}_{i-1} \cap P \neq \emptyset$ for all $i \in \{-t+1,\ldots,k\}$. Consequently, each strip contains a point of P.

In order to prove that $\text{SLAB}(R,p)$ is connected, we describe below a strategy that does not quite work but, as we show later, can be fixed.

Let $\text{STRIP}_i \cap P = P_i$. By Condition (s-iii), $P_i \neq \emptyset$ for any $i \in \{-t,\ldots,k\}$. For a strip STRIP_i, let p_i denote the rightmost point in it. Let us assume for now that for each $i \in \{-t,\ldots,k\}$, and each point $q \in P_i$, there is a path from q to p_i that lies entirely in $\textsc{Strip}_i$.[9] Now, consider an arbitrary point $q \in \textsc{Slab}(R,p)$. By Condition (s-ii) each point in $P \cap \textsc{Slab}(R,p)$ is contained in at least one strip. Therefore, $q \in \textsc{Strip}_i$ for some $i \in \{-t,\dots,k\}$. By our assumption, there is a path π_i^1 from q to p_i that lies entirely in $\textsc{Strip}_i$. If $i \geq 0$ (a symmetric argument works when $i < 0$), since Condition (s-iii) implies consecutive strips intersect at a point in P, there is a path π_i^2 from p_i to a point $q' \in P_i \cap P_{i-1}$ that lies entirely in $\textsc{Strip}_i$. Again, by our assumption, there is a path π_{i-1}^1 from q' to p_{i-1} that lies entirely in $\textsc{Strip}_{i-1}$. Repeating the argument above with $i-1, i-2, \dots$, until $i = 0$, and concatenating the paths $\pi_i^1, \pi_i^2, \pi_{i-1}^1, \dots$, we obtain a path π from q to p, each sub-path of which is a path from a point in a strip to the rightmost point in that strip such that each point in the path lies entirely in the strip. By Condition (s-i), $\textsc{Strip}_i \subseteq \textsc{Slab}(R,p)$ for each $i \in \{-t,\dots,k\}$. Therefore, π lies entirely in $\textsc{Slab}(R,p)$. Since q was arbitrary, this implies that $\textsc{Slab}(R,p)$ is connected.

Consider a slab $\textsc{Slab}(R,p)$ corresponding to a rectangle $R \in \mathcal{R}$ and a point $p \in P \cap R$. The strips corresponding to $\textsc{Slab}(R,p)$ are defined as follows: Let s denote the open segment of ℓ_p between p and $y_+(\textsc{Slab}(R,p))$ of ℓ_p, the vertical line through p. Let $\mathcal{R}_s = (R_0,\dots,R_h)$ be the rectangles in $\textsc{Active}(s)$ ordered by their upper sides i.e., $y_+(R_i) < y_+(R_j)$, for $0 \leq i < j \leq h$. Similarly, let s' denote the open segment of ℓ_p between p and $y_-(\textsc{Slab}(R,p))$ and let $\mathcal{R}_{s'} = (R'_0,\dots,R'_{h'})$ denote the rectangles in $\textsc{Active}(s')$ ordered by their lower sides $y_-(R'_i) > y_-(R'_j)$ for $0 \leq i < j \leq h'$.

We define $\textsc{Strip}_0 = \textsc{Slab}(R,p)[y_-(R'_0), y_+(R_0)]$, if $\textsc{Active}(s) \neq \emptyset$ and $\textsc{Active}(s') \neq \emptyset$. If $\textsc{Active}(s) = \emptyset$, we set $y_+(\textsc{Strip}_0) = y_+(\textsc{Slab}(R,p))$. Similarly, if $\textsc{Active}(s') = \emptyset$, we set $y_-(\textsc{Strip}_0) = y_-(\textsc{Slab}(R,p))$. We set $p_0 = p$. Having defined $\textsc{Strip}_0$, we set $\mathcal{R}_s = \mathcal{R}_s \setminus R_0$ and $\mathcal{R}_{s'} = \mathcal{R}_{s'} \setminus R'_0$.

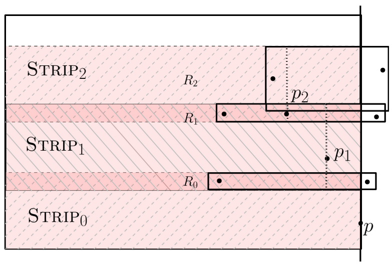


Figure 2 The figure shows the construction of the strips $\textsc{Strip}_0$, $\textsc{Strip}_1$ and $\textsc{Strip}_2$. The vertical line segment through p_i, $i \in \{1,2\}$ shows that p_i is the rightmost point among the points in the strip i.

For $i > 0$, having constructed $\textsc{Strip}_j$ for $j = 0,\dots,i-1$, we do the following while $\mathcal{R}_s \neq \emptyset$: Let $S_i = \operatorname{argmin}_{R' \in \mathcal{R}_s} y_-(R')$, and let $y_- = y_-(S_i)$. Let $R_i = \operatorname{argmin}\{y_+(R') : R' \in \mathcal{R}_s : y_-(R') > y_-\}$, and let $y_+ = \min\{y_+(\textsc{Slab}(R,p)), y_+(R_i)\}$. Set $y_-(\textsc{Strip}_i) = y_-$ and $y_+(\textsc{Strip}_i) = y_+$. Let $p_i = \operatorname{argmax}\{x(p') : p' \in P \cap \textsc{Slab}(R,p) : y_- < y(p') < y_+\}$. Note that p_i exists since $S_i \in \textsc{Active}(s)$. Set $\mathcal{R}_s = \mathcal{R}_s \setminus \{R' : y_-(R') < y_-(R_i)\}$.

For $i < 0$, the construction is symmetric. Having constructed $\textsc{Strip}(R,p,j)$ for $j = 0,-1,\dots,-i+1$, we do the following until $\mathcal{R}_{s'} = \emptyset$. Let $S'_i = \operatorname{argmax}_{R' \in \mathcal{R}_{s'}} y_+(R')$, and let $y_+ = y_+(S'_i)$. Let $R'_i = \operatorname{argmax}\{y_-(R') : R' \in \mathcal{R}_{s'}, y_+(R') < y_+\}$. Let $y_- = \max\{y_-(R'_i), y_-(\textsc{Slab}(R,p))\}$. Set $y_-(\textsc{Strip}_i) = y_-$ and $y_+(\textsc{Strip}_i) = y_+$. Let $p_i = \operatorname{argmax}\{x(p') : p' \in P \cap \textsc{Slab}(R,p), y_- < y(p') < y_+\}$. Again, p_i exists since $S'_i \in \textsc{Active}(s')$. Set $\mathcal{R}_{s'} = \mathcal{R}_{s'} \setminus \{R' \in \mathcal{R}_{s'} : y_+(R') > y_+(R'_i)\}$. Figure 2 illustrates the construction of the strips.

[9] This assumption is incorrect but will be remedied later.

▶ **Proposition 1.** *For* $i \in \{-t, \ldots, k\}$,

$$y_-(\text{Lb}(p_i)) \leq y_-(\text{Strip}_i) < y_+(\text{Strip}_i) \leq y_+(\text{Ub}(p_i))$$

Proof. Fix $i \in \{-t, \ldots, k\}$ and assume $i \geq 0$. For $i < 0$, the proof is symmetric. Since $p_i \in \text{Strip}_i$ and by the definition of the lower barrier, $y_+(\text{Lb}(p_i)) < y(p_i) < y_+(\text{Strip}_i)$. If $y_-(\text{Lb}(p_i)) > y_-(\text{Strip}_i)$, since $\text{Lb}(p_i) \in \mathcal{R}_s$ and $y_+(\text{Strip}_i) \leq \min\{y_+(R') \in \mathcal{R}_s : R' \in \mathcal{R}_s \text{ and } y_-(R') > y_-(\text{Strip}_i)\}$, it implies $y_+(\text{Strip}_i) \leq y_+(\text{Lb}(p_i))$, contradicting $p_i \in \text{Strip}_i$.

Now we argue about $\text{Ub}(p_i)$. If $y_+(\text{Ub}(p_i)) > y_+(\text{Slab}(R,p))$, since $y_+(\text{Slab}(R,p)) \geq y_+(\text{Strip}_i)$, we have $y_+(\text{Ub}(p_i)) \geq y_+(\text{Strip}_i)$. Otherwise, we have $\text{Ub}(p_i) \in \mathcal{R}_s$. Since $p_i \in \text{Strip}_i$ and by definition of the upper barrier, we have $y_-(\text{Ub}(p_i)) > y(p_i) > y_-(\text{Strip}_i)$. Since $y_+(\text{Strip}_i) \leq \min\{y_+(R') : R' \in \mathcal{R}_s \text{ and } y_-(R') > y_-(\text{Strip}_i)\}$, it follows that $y_+(\text{Strip}_i) \leq y_+(\text{Ub}(p_i))$. ◀

▶ **Lemma 2.** *The strips constructed as above satisfy the following conditions: (i)* $\text{Strip}_i \subseteq \text{Slab}(R,p)$ *for each* $i \in \{-t, \ldots, k\}$*. (ii)* $\text{Slab}(R,p) = \cup_{i=-t}^{k}\text{Strip}_i$*, and (iii)* $\text{Strip}_i \cap \text{Strip}_{i-1} \cap P \neq \emptyset$ *for all* $i \in \{-t+1, \ldots, k\}$.

Proof. Item (i) and (ii) follow directly by construction. For (iii), note that adjacent strips contain a piece of an active rectangle and hence their intersection contains a point of P. ◀

Unfortunately, our assumption that every point in a strip has a path to its rightmost point in the strip is not correct. To see this, consider a strip that is pierced by a rectangle R', whose intersection with the strip does not contain a point of P. Therefore, a point in the strip that lies to the left of R' cannot have a path to p_i that lies in the strip unless some of its edge is allowed to pierce R'. In order to remedy this situation, we introduce the notion of a *corridor*. A corridor corresponding to a strip is a region of $\text{Slab}(R,p)$ that contains all points in the strip, and such that each point in the strip has a path to the rightmost point in it that lies entirely in the corridor. Since each corridor lies in $\text{Slab}(R,p)$, the proof strategy can be suitably modified to show that $\text{Slab}(R,p)$ is connected.

We now define the corridors associated with each strip. Recall that $G = (P, E)$ is the graph constructed by Algorithm 1. For a point $q \in P$, recall that the neighbors of q in G that lie to its left are called its *left-adjacent* points. If q lies on a path π, and q' is the left-adjacent point of q on π, then we say that q' is left-adjacent to q on π. We start with the following proposition that will be useful in constructing the corridors.

▶ **Proposition 3.** *For a point* $q \in P$*, if* $(q_1, \ldots, q_r, q_{r+1}, \ldots, q_s)$ *is the sequence of its left-adjacent points in* G *s.t.* $y(q_1) > \ldots > y(q_r) > y(q)$ *and* $y(q_{r+1}) < \ldots < y(q_s) < y(q)$*. Then, for* $1 \leq i < j \leq r$*,* $x(q_i) > x(q_j)$*, and for* $r+1 \leq i < j \leq s$*,* $x(q_i) < x(q_j)$*.*

Proof. This follows directly from the fact that each edge in G is Delaunay. ◀

For a strip Strip_i, we define its corresponding corridor Corridor_i as follows: The corridor is the region of $\text{Slab}(R,p)$ bounded by two paths: an *upper path* π_i^u, and a *lower path* π_i^ℓ, defined as follows.

The upper path $\pi_i^u = (q_0, q_1, q_2, \ldots,)$ is constructed by starting with $j = 0, q_0 = p_i$, and repeating (1) set $q_{j+1} \leftarrow q'$ where q' has the highest $y(q')$ among the left-adjacent points of q_j. (2) $j \leftarrow j+1$. We stop when we cannot find such a q' for the current q_j in G that lies in $\text{Slab}(R,p)$, where we complete the path by following the edge to the left-adjacent vertex q_{j+1} of q_j with highest y-coordinate. Thus, the last vertex of π_i^u possibly does not lie in $\text{Slab}(R,p)$.

The lower path $\pi_i^\ell = (q'_0, \ldots,)$ is constructed similarly. For $j = 0$, set $q'_0 = p_i$, and repeating (1) set $q'_{j+1} \leftarrow q'$, where q' has the *lowest* $y(q')$ among the left-adjacent points of q'_j. We stop when we cannot find such a q' in $\textsc{Slab}(R, p)$ for the current q'_j, where we complete the path by following the edge from q'_j its left-adjacent vertex q'_{j+1} with smallest y-coordinate. Thus, the last vertex of π_i^ℓ possibly does not lie in $\textsc{Slab}(R, p)$.

$\textsc{Corridor}_i$ is the region of $\textsc{Slab}(R, p)$ that lies between the upper and lower paths π_i^u and π_i^ℓ. Figure 3 shows a corridor corresponding to a strip. We start with some basic observations about the corridors thus constructed.

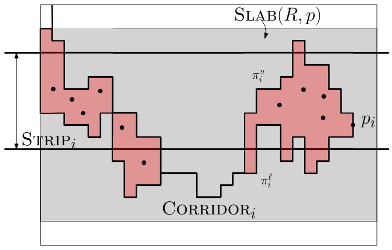


Figure 3 The figure above shows the strip $\textsc{Strip}_i$, the slab $\textsc{Slab}(R, p)$ in grey, and the corridor $\textsc{Corridor}_i$ as the region shaded in red between π_i^u and π_i^ℓ.

▶ **Proposition 4.** *For $i \in \{-t, \ldots, k\}$, π_i^u and π_i^ℓ are x-monotone.*

Proof. This follows directly by construction since at each step we augment the path by adding to it, the left-adjacent neighbor to the current vertex of the path. ◀

The graph G constructed by Algorithm 1 do not cross. We say that two paths π_1 and π_2 in G *cross* if there is an x-coordinate at which π_1 lies above π_2, and an x-coordinate at which π_2 lies above π_1.

▶ **Proposition 5.** *For $i \in \{-t, \ldots, k\}$, π_i^l does not lie above π_i^u, and π_i^ℓ and π_i^u do not cross.*

Proof. Let $\pi_i^u = (q_0, q_1, \ldots, q_r)$ and $\pi_i^\ell = (q'_0, \ldots, q'_s)$, where $q_0 = q'_0 = p_i$. Since q_1 is the left-adjacent of p_i in $\textsc{Slab}(R, p)$ with highest y-coordinate and q'_1 is the left-adjacent point of p_i with lowest y-coordinate, $y(q'_1) \leq y(q_1)$. Thus, π_i^ℓ does not lie above π_i^u at $x(p_i)$. If at some x-coordinate x', π_i^ℓ lies above π_i^u, then the paths must have crossed to the right of x'.

Let $q_i = q'_j = q$ be a point of P common to π_i^u and π_i^ℓ lying to the left of p_i. Again, since $q_{i+1} = \arg\max\{y(q'') : q'' \in \textsc{Slab}(R, p) \cap P, x(q'') < x(q), \{q, q''\} \in E(G)\}$, and $q'_{j+1} = \arg\min\{y(q'') : q'' \in \textsc{Slab}(R, p) \cap P, x(q'') < x(q), \{q, q''\} \in E(G)\}$, it follows that $y(q'_{j+1}) \leq y(q_{i+1})$. Hence, the paths do not cross, and since π_i^ℓ does not lie above π_i^u at $x(p_i)$, it does not do so at any x-coordinate to the left of p_i either. ◀

Recall that the left-neighbor of a point q in a set P' is the point $p' = \text{argmax}_{p'' \in P'} x(p'') < x(q)$. The right-neighbor is defined similarly. Note that the left and right neighbors are defined geometrically, and they may not be adjacent to q in the graph G. For a point $q \in P$ and $i \in \{-t, \ldots, k\}$, we let r_0 and r_1 denote respectively, the left- and right-neighbors of q on π_i^u. Similarly, we let r'_0 and r'_1 denote respectively, the left- and right-neighbors of q on π_i^ℓ.

▶ **Proposition 6.** *For $i \in \{-t, \ldots, k\}$ and $q \in P$, if q lies above π_i^u, then $y(q) > \max\{y(r_0), y(r_1)\}$. Similarly, if q lies below π_i^ℓ, then $y(q) < \min\{y(r'_0), y(r'_1)\}$.*

Proof. We assume q lies above π_i^u. The other case follows by an analogous argument. By Proposition 4, since π_i^u is x-monotone, it follows that r_0 and r_1 are consecutive along π_i^u, and thus, r_0r_1 is a valid Delaunay edge in G. If either $y(r_0) > y(q) > y(r_1)$, or $y(r_0) < y(q) < y(r_1)$, then it contradicts the fact that r_0r_1 is Delaunay. Hence, $y(q) > \max\{y(r_0), y(r_1)\}$ as q lies above π_i^u. ◀

We now show that the corridors constructed satisfy the required conditions. The first condition below, follows directly from construction.

▶ **Lemma 7.** *For $i \in \{-t, \dots, k\}$, $\textsc{Corridor}_i \subseteq \textsc{Slab}(R, p)$.*

Proof. Follows directly by construction. ◀

Next, we show that for each strip, its corresponding corridor contains all its points, that is all points in $P \cap \textsc{Strip}_i$ are contained between the upper and lower paths of $\textsc{Corridor}_i$. Before we do that, we need the following two technical statements.

▶ **Proposition 8.** *Let $q, q' \in P$, with $x(q) < x(q')$ s.t. qq' is Delaunay. If $qq' \notin E(G)$, then either (i) $h(qq')$ pierces a rectangle, or crosses an existing edge, or (ii) $v(qq')$ pierces a rectangle. In particular, $v(qq')$ does not cross an existing edge.*

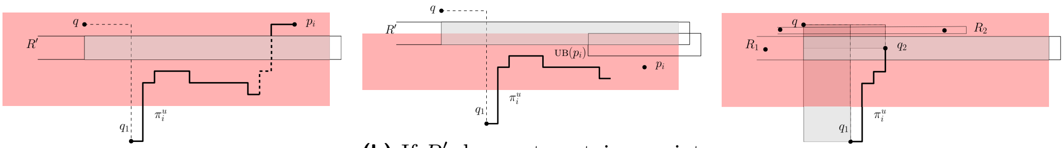
Proof. The points in P are processed by Algorithm 1 in increasing order of their x-coordinates, and when a point is being processed, we add edges of type $\{\urcorner, \lrcorner\}$ to points to its left. Therefore, while processing q', no edge from points of P to the right of q' have been added. Hence, $v(qq')$ does cross an existing edge. ◀

▶ **Lemma 9.** *For $i \in \{-t, \dots, k\}$, let $q \in \textsc{Slab}(R, p) \cap P$ s.t. q lies above π_i^u. Let q_1 be the right-neighbor of q on π_i^u. If qq_1 is Delaunay but not valid, then $v(qq_1)$ pierces a rectangle. In particular, $h(qq_1)$ does not pierce a rectangle or cross an edge. Similarly, let $q' \in \textsc{Slab}(R, p) \cap P$ s.t. q' lies below π_i^ℓ, q'_1 is the right-neighbor of q' on π_i^ℓ. If $q'q'_1$ is Delaunay but not valid, then $v(q'q'_1)$ pierces a rectangle. In particular $h(q'q'_1)$ does not pierce a rectangle or cross an edge.*

Proof. We prove the case when q lies above π_i^u. The other case follows by an analogous argument. Since qq_1 is not valid, either the horizontal segment of qq_1 pierces a rectangle, or crosses an existing edge; or $v(qq_1)$ pierces a rectangle since by Proposition 8, $v(qq_1)$ does not cross an existing edge.

Let q_0 be left-adjacent to q_1 on π_i^u. By Proposition 6, $y(q) > \max\{y(q_0), y(q_1)\}$. Hence qq_1 is of type $\urcorner$. Suppose $h(qq_1)$ pierces a rectangle or crosses an edge of type $\urcorner$. Then, there is a point z that lies below the $h(qq_1)$. But, z cannot lie below the $h(q_0q_1)$, as that contradicts the fact that q_0q_1 is valid. Hence, z lies above $h(q_0q_1)$. This implies that z lies either in $R(q_0q_1)$ (if $y(q_0) < y(q_1)$), or z lies in $R(qq_1)$. If $z \in R(q_0q_1)$, it contradicts the fact that q_0q_1 is Delaunay. Also, $z \notin R(qq_1)$, as qq_1 is Delaunay by assumption. Therefore, $h(qq_1)$ does not pierce a rectangle, or crosses an edge of type $\urcorner$. If $h(qq_1)$ crossed an edge e of type $\lrcorner$, then either qq_1 is not Delaunay, violating our assumption, or e is not Delaunay, a contradiction. ◀

▶ **Lemma 10.** *For $i \in \{-t, \dots, k\}$, $P_i \subseteq \textsc{Corridor}_i$, where $P_i = P \cap \textsc{Strip}_i$.*



(a) R' cannot be empty if p_i is above R'.

(b) If R' does not contain a point of P between $x(q)$ and $x_+(R)$, then $y(q) > y_+(\textsc{Strip}_i)$.

(c) Since qq_1 and qq_2 are Delaunay, then R_2 pierces $v(qq_1)$.

Figure 4 The three cases in the proof showing that $P_i \subseteq \textsc{Corridor}_i$.

Proof. Suppose $P_i \setminus \textsc{Corridor}_i \neq \emptyset$. By Proposition 5, any such point either lies above π_i^u or below π_i^ℓ. We assume the former. The latter follows by an analogous argument. Let $P_i' = \{q \in \textsc{Slab}(R,p) : y(q) < y_+(\textsc{Strip}_i)$ and q lies above $\pi_i^u\}$. It suffices to show that $P_i' = \emptyset$. For the sake of contradiction, suppose $P_i' \neq \emptyset$.

We impose the following partial order on P_i: for $a, a' \in P_i$, $a \prec a' \Leftrightarrow x(a) > x(a') \wedge y(a) < y(a')$. Let q be a minimal element in P_i' according to $\prec$. In the following, when we refer to a minimal element, we impicitly assume this partial order.

We show that there is a valid Delaunay edge between q and a point q' on π_i^u. By assumption, q lies above π_i^u. Let q_0 and q_1 denote, respectively, the left- and right-neighbors of q on π_i^u.

Since q is minimal in P_i', qq_1 is Delaunay. By Proposition 6, it follows that $y(q) > y(q_0)$ and $y(q) > y(q_1)$. Since q is not left-adjacent to q_1 on π_i^u, it implies qq_1 is not valid.

Since qq_1 is Delaunay but not valid, by Lemma 9, $v(qq_1)$ pierces a rectangle. Let $\mathcal{R}'$ denote the set of rectangles pierced by qq_1. Suppose $\exists R' \in \mathcal{R}'$ s.t. $R'[x(q_1), \min\{x(p), x_+(R')\}] \cap P = \emptyset$. Then, we call R' a *bad* rectangle. Otherwise, we say that R' is *good*. Now we split the proof into two cases depending on whether $\mathcal{R}'$ contains a bad rectangle or not. In the two cases below, we use Proposition 4 that π_i^u is x-monotone.

Case 1. $\mathcal{R}'$ contains a bad rectangle. Let $R' \in \mathcal{R}'$ be a bad rectangle. First, observe that $x_+(R') > x(p)$ as otherwise, R' is not pierced by $v(qq_1)$. Now, suppose $y(p_i) > y_+(R')$, where p_i is the rightmost point in $\textsc{Strip}_i$. We have that $x(q_1) < x(p_i)$, both p_i and q_1 lie on π_i^u and, π_i^u is x-monotone. But, this implies π_i^u pierces R'. But this is impossible as by construction π_i^u consists of valid Delaunay edges. Hence, since R' is bad, we can assume that $y(p_i) < y_-(R')$. But the definition of the upper barrier implies that $y_+(R') \geq y_+(\textsc{Ub}(p_i))$. Since $v(qq_1)$ pierces R', it implies $y(q) > y_+(R')$, and hence $y(q) > y_+(\textsc{Ub}(p_i))$. But, this contradicts the assumption that $q \in \textsc{Strip}_i$, since by Proposition 1, $y_+(\textsc{Ub}(p_i)) \geq y_+(\textsc{Strip}_i)$ and hence $y(q) > y_+(\textsc{Strip}_i)$.

Case 2. All rectangles in $\mathcal{R}'$ are good. Let $R_1 = \arg\max\{y_-(R') : R' \in \mathcal{R}'\}$. Let q_2 be the leftmost point in R_1 s.t. $x(q_2) > x(q_1)$. Since q_2 lies to the right, and below q, $q_2 \prec q$. Since q is a minimal element in P_i', it implies that q_2 lies on or below π_i^u. We claim that q_2 cannot lie below π_i^u. Suppose it did. Let q_2' be the left-neighbor of q_2 on π_i^u. Then, $x(q_2') < x(q_2)$. Since q_2 is the leftmost point of R_1 to the right of $v(qq_1)$, then either $y(q_2') < y_-(R_1)$, or $y(q_2') > y_+(R_1)$. However, in either case, we obtain that π_i^u must cross R_1 between q_2 and q_1, which implies that π_i^u pierces R_1, as π_i^u is x-monotone, and the edges in π_i^u are of the form $\{\urcorner, \lrcorner\}$, a contradiction.

Since q is minimal, qq_2 is Delaunay. By Lemma 9, the only reason qq_2 is not valid is that $v(qq_2)$ pierces a rectangle. But, any such rectangle R_2 is also pierced by $v(qq_1)$, as qq_2 is Delaunay. But, this implies $y_-(R_2) > y_-(R_1)$, contradicting the choice of R_1. Therefore, qq_2 is a valid Delaunay edge. Now, the only reason that q is not the left-adjacent point of q_2

on π_i^u then, is that q_2', the left-adjacent point of q_2 on π_i^u lies in $\textsc{Slab}(R,p)$, but above q, i.e., $y(q_2') > y(q)$, as the construction of π_i^u dictates that we choose the left-adjacent point with highest y-coordinate that lies in $\textsc{Slab}(R,p)$. Showing that this leads to a contradiction completes the proof.

So suppose $y(q_2') > y(q)$, then $y(q_2') > y_+(R_1)$. Further, by Proposition 3, $x(q_2') > x(q)$. Again this implies the x-monotone curve π_i^u cannot contain both q_2' and q_1 without piercing R_1. Hence, $y(q_2') < y_-(R_1) < y(q)$, but this contradicts the choice of q_2' as the left-adjacent point of q_2 on π_i^u since qq_2 is a valid Delaunay edge with $y(q_2') < y(q_2) < y(q)$. See Figure 4 for the different cases in this proof. ◀

The key property of a corridor is that if the upper or lower path of a corridor crosses a rectangle R', then there must be a point of $R' \cap P$ that lies on that path of the corridor. Using this, we can show that any point in the strip has an adjacent point to its right in G that lies in the corridor. This implies that every point in a strip has a path to the rightmost point in the strip that lies entirely in its corresponding corridor.

▶ **Lemma 11.** *For each $q \in \textsc{Corridor}_i$, there is a path $\pi(q,p_i)$ between q and p_i that lies in $\textsc{Corridor}_i$, where $p_i \in P_i$ is the rightmost point in $\textsc{Strip}_i$.*

Proof. If q lies on the upper path π_i^u or the lower path π_i^ℓ defining $\textsc{Corridor}_i$, the lemma is immediate. So we can assume by Proposition 5 that q lies below π_i^u, and above π_i^ℓ. Suppose the lemma is false. Let q be the rightmost point of $\textsc{Corridor}_i$ that does not have a path to p_i lying in $\textsc{Corridor}_i$. To arrive at a contradiction, it is enough to show that q is adjacent to a point $q' \in \textsc{Corridor}_i$ that lies to the right of q.

Starting from ℓ_q, the vertical line through q, sweep to the right until the first point r that lies on both π_i^u and π_i^ℓ. Such a point exists since p_i lies on both π_i^u and π_i^ℓ. Let Q_i' denote the set of points in $\textsc{Corridor}_i$ whose x-coordinates lie between $x(q)$ and $x(r)$. This set is non-empty as it contains q and r. Hence, either $Q_i^+ = \{q' \in Q_i' : y(q') > y(q)\} \neq \emptyset$, or $Q_i^- = \{q' \in Q_i' : y(q') < y(q)\} \neq \emptyset$. If both are non-empty, let Q_i denote the set that contains a point with smallest x-coordinate. Otherwise, we let Q_i denote the unique non-empty set. Assume $Q_i = Q_i^+$. An analgous argument holds when $Q_i = Q_i^-$.

Define a partial order on Q_i, where for $a, b \in Q_i$, $a \prec b \Leftrightarrow x(a) < x(b)$ and $y(a) < y(b)$. Let $Q_i^{\min} = (q_1, \ldots, q_t)$ denote the sequence of minimal elements of Q_i ordered linearly such that $y(q_k) > y(q_j)$ for $k < j$. It follows that $x(q_k) < x(q_j)$ for $k < j$. Observe that qq_i is Delaunay for $i = 1, \ldots, t$ by the minimality of q_i. Our goal is to show that qq_i is a valid Delaunay edge for some $i \in \{1, \ldots, t\}$. We start with the following claim that $v(qq_t)$ and $h(qq_1)$ do not pierce a rectangle in $\mathcal{R}$, or cross an edge of G.

▷ Claim 12. $h(qq_1)$ does not pierce a rectangle in $\mathcal{R}$ or cross an edge of G, and $v(qq_t)$ does not pierce a rectangle in $\mathcal{R}$ or cross an edge of G.

Proof. Suppose $h(qq_1)$ pierced a rectangle R'. Since π_i^u consists of valid Delaunay edges, and the choice of Q_i, R' contains a point a that lies in $\textsc{Corridor}_i$. Since $h(qq_1)$ pierces R', $x(q) < x(a) < x(q_1)$. If $y(a) < y(q_1)$, then it contradicts the fact that q_1 is minimal, and if $y(a) > y(q_1)$, it contradicts the fact that q_1 is the minimal element with highest y-coordinate. A similar argument shows that $h(qq_1)$ does not cross an edge of G.

If $v(qq_t)$ pierced a rectangle $R' \in \mathcal{R}$, then R' has a point to the right of $v(qq_t)$. Further, by Proposition 4, π_i^u and π_i^ℓ are x-monotone paths and by construction, they consist of valid Delaunay edges meeting at r. If $r = q_t$ and $v(qq_t)$ pierced a rectangle, since qq_t is Delaunay, and the edges are of type $\{⌉, ⌋\}$, it implies that the left-adjacent point of r on π_i^u or π_i^ℓ is not a valid Delaunay edge. Hence, we can assume $q_t \neq r$. Again, since the edges of π_i^u and

π_i^ℓ are valid Delaunay edges, it implies that R' has a point a s.t. $x(q_t) < x(a) \leq x(r)$, and a lies in Corridor$_i$. Since $v(qq_t)$ pierces R', $y(a) < y(q_t)$. But this contradicts the fact that q_t is the point in $Q_i^{\min}$ with the smallest y-coordinate. Therefore, $v(qq_t)$ can not pierce any rectangle. Since qq_t is Delaunay, by Proposition 8, $v(qq_t)$ does not cross an edge of G. ◁

We now define a point x_1 on the x-axis and a point y_1 on the y-axis as follows:

$$x_1 = \min \{\{x_+(R') : R' \text{ pierced by } h(qq_t)\}, \{v(e) : e \text{ crosses } h(qq_t)\}\}$$
$$y_1 = \min \{\{y_+(R') : R' \text{ pierced by } v(qq_1)\}, \{h(e) : e \text{ crosses } v(qq_1)\}\}$$

By Claim 12 and the assumption that qq_t and qq_1 are not valid edges, it follows that x_1 and y_1 exist. We argue when x_1 and y_1 correspond to $x_+(R')$ and $y_+(R'')$, respectively for rectangles $R', R'' \in \mathcal{R}$. If they were instead defined by the vertical/horizontal side of edges, the arguments are similar.

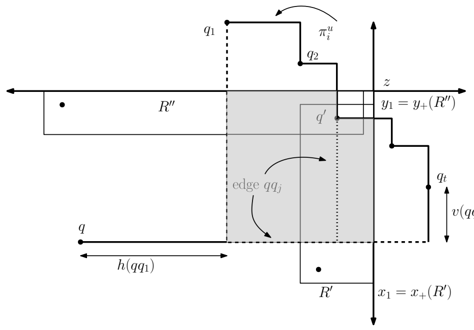


Figure 5 The edge qq' is a valid Delaunay edge.

Observe that $x_-(R'') < x(q)$, while $x_-(R') > x(q)$, and $y_-(R'') > y(q)$ and $y_-(R') < y(q)$. Now, from the fact that the rectangles are non-piercing, it implies that either $x_+(R'') < x_1$ or $y_+(R') < y_1$. Suppose wlog, the former is true. Since R' is pierced by $h(qq_t)$ and π_i^u consists of valid Delaunay edges, there are points in R' that lie in Corridor$_i$, and these points lie below y_1.

Let z denote the intersection of the vertical line through x_1 and the hoizontal line through y_1. By the argument above, the rectangle with diagonal qz contains points of P, and hence a point $q' \in Q_i^{\min}$. We claim that qq' is a valid Delaunay edge. To see this, note that $h(qq')$ does not pierce a rectangle in $\mathcal{R}$ as such a rectangle contradicts the definition of x_1. If $v(qq')$ pierced a rectangle, such a rectangle $\tilde{R}$ must have $y_+(\tilde{R}) < y_1$, as qq' is Delaunay. This contradicts the choice of y_1. Therefore, qq' is a valid Delaunay edge. ◀

The lemma below follows the description in the proof strategy at the start of this section.

▶ **Lemma 13.** *For a rectangle R and point $p \in R$, after Algorithm 1 has processed point p, the points in* Slab(R,p) *induce a connected subgraph, all of whose edges lie in* Slab(R,p).

Proof. Let $G[\text{Slab}(R,p)]$ denote the induced subgraph of G on the points in Slab(R,p). By Condition (ii) of Lemma 2, since Slab$(R,p) \subseteq \cup_{i=-t}^{k}\text{Strip}_i$, each point in $P \cap \text{Slab}(R,p)$ is contained in $\cup_{i=-t}^{k}\text{Strip}_i$. If the statement of the lemma does not hold, consider an extremal strip, i.e., the smallest positive index, or largest negative index of a strip such that it contains a point q that does not lie in the connected component of $G[\text{Slab}(R,p)]$ containing p. Assume without loss of generality that $i \geq 0$. An analogous argument holds if $i < 0$. By Lemma 10, $q \in$ Corridor$_i$, and by Lemma 11, q has a path π_1 to p_i, the rightmost point in $\textsc{Corridor}_i$ that lies entirely in $\textsc{Corridor}_i$. By Condition (ii) of Lemma 2, there is a point $q' \in \textsc{Strip}_i \cap \textsc{Strip}_{i-1} \cap P$. By Lemma 10, $q' \in \textsc{Corridor}_i$, and by Lemma 11, there is a path π_2 between q' and p_i. Since $q' \in \textsc{Strip}_{i-1}$, q' lies in the same connected component as p in $G[\textsc{Slab}(R,p)]$, and hence there is a path π' from q' to p in $G[\textsc{Slab}(R,p)]$. Concatenating π_1, π_2 and π' we obtain a path π from q to p that lies in $\textsc{Slab}(R,p)$. ◀

We now argue that if p is the rightmost point in a rectangle R, then $\textsc{Piece}(R,p)$ consists of a single slab.

▶ **Lemma 14.** *If p is the last point in R according to the x-coordinates of the points, then $\textsc{Piece}(R,p)$ consists of a single slab.*

Proof. Assume for the sake of contradiction that $\textsc{Upb}(R,p)$ exists. By definition of $\textsc{Upb}(R,p)$, there are two points $a,b \in \textsc{Upb}(R,p)$, such that $x(a) < x_-(R) < x(p) < x_+(R) < x(b)$, as p is the last point in R. But this implies $\textsc{Upb}(R,p)$ is pierced by R, a contradiction. Therefore, $\textsc{Upb}(R,p)$ does not exist. Similarly, $\textsc{Lpb}(R,p)$ does not exist, and hence $\textsc{Piece}(R,p)$ consists of a single slab. ◀

▶ **Theorem 15.** *Algorithm 1 constructs a planar support.*

Proof. By construction, the edges of the graph G constructed by Algorithm 1 are valid Delaunay edges of type $\{\urcorner, \lrcorner\}$. To obtain a plane embedding, we replace each edge $e = \{p,q\}$ by the diagonal of the rectangle $R(pq)$ joining p and q. We call these the *diagonal edges*. It is clear that no diagonal edge pierces a rectangle. If two diagonal edges cross, then it is easy to check that either the corresponding edges cross, or they are not Delaunay. For a rectangle $R \in \mathcal{R}$, let p be the last point in R. Lemma 14 implies that there is only one slab, namely R, and Lemma 13 implies $\textsc{Slab}(R,p)$ is connected. Since R was arbitrary, this implies Algorithm 1 constructs a support. ◀

4.2 Implementation

In this section, we show that Algorithm 1 can be implemented to run in $O(n \log^2 n + (m + n)\log m)$ time with appropriate data structures, where $|\mathcal{R}| = m$, and $|P| = n$. At any point in time, our data structure maintains a subset of points that lie to the left of the sweep line ℓ. It also maintains for each rectangle R intersecting ℓ, the interval $[y_-(R), y_+R]$ corresponding to R. When the sweep line arrives at the left side of a rectangle, the corresponding interval is inserted into the data structure. The interval is removed from the data structure when the sweep line arrives at the right side of the rectangle. Similarly, whenever we sweep over a point p, we insert it into the data structure. In addition, we do the following when the sweep line arrives at a point p:

1. Find the upper and lower barriers at p.
2. Query the data structure to find the set Q of points q which *i*) lie to the left of p and between the upper and lower barriers at p (orthogonal range query) so that *ii*) qp is a Delaunay edge.
3. We add the edge qp for every $q \in Q$ to our planar support. For each edge we add, we remove the points in the data structure that are *occluded* by the edges. These are the points whose y-coordinates lie in the range corresponding to the vertical side of the L-shape for qp.

Our data structure is implemented by combining three different existing data structures. For Step 1, we use a balanced binary search tree $\mathcal{T}_1^u$ augmented so that it can answer range minima or maxima queries. For any rectangle R intersecting the sweep line ℓ, let

(y_1, y_2) denote the interval corresponding to the projection of R on the y-axis. $\mathcal{T}_1^u$ stores the key-value pair (y_1, y_2) with y_1 as the key and y_2 as the value. To find the upper barrier at a point $p = (x, y)$ we need to find the smallest value associated with keys that are at least x. If we augment a standard balanced binary search tree so that at each node we also maintain the smallest value associated with the keys in the subtree rooted at that node, such a query takes $O(\log m)$ time. An analogous search tree $\mathcal{T}_1^b$ is used to find the lower barrier at any point.

To implement Step 2, we use a dynamic data structure $\mathcal{T}_2^b$ due to Brodal [10] which maintains a subset of the points to the left of ℓ and can report points in any query rectangle Q that are not dominated by any of the other points in time $O(\log^2 n + k)$ where k is the number of reported points. We say that a point u is dominated by a point v if both x and y coordinates of u are smaller than those of v. The data structure also supports insertions or deletions of points in $O(\log^2 n)$ time. When the sweep line arrives at a point p, we can use $\mathcal{T}_2^b$ to find all points q that lie to the left of p and below p so that the edge qp is a Delaunay edge (as qp of shape $\lrcorner$ is Delaunay iff there is no other point in the range below and to the left of p that dominates q). An analogous data structure $\mathcal{T}_2^u$ is used to find the points q which lie above and to the left of p so that qp (of shape $\urcorner$) is Delaunay.

To implement Step 3, we use a dynamic 1D range search data structure $\mathcal{T}_3$ which also stores a subset of the points to the left of ℓ, supports insertions and deletions in $O(\log n)$ time and can report in $O(\log n + k)$ time the subset of stored points that lie in a given range of y-coordinates (corresponding the vertical side of each added edge), where k is the number of points reported. The points identified are removed from $\mathcal{T}_2^u, \mathcal{T}_2^b$ and $\mathcal{T}_3$.

By the correctness of Algorithm 1 proved in Section 4.1, at any point in time, the current graph is a support for the set of rectangles that lie completely to the left of the sweep line. Thus, if the sweep line ℓ is currently at a point p and q is a point to the left of ℓ, the only rectangles that qp may discretely pierce are those that intersect ℓ. A simple but important observation is that if qp is Delaunay then qp pierces a rectangle iff the vertical portion of L-shape forming the edge pq pierces the rectangle. To see this note that the horizontal portion of the L-shape cannot pierce any rectangle since such a rectangle would not intersect ℓ. The L-shape also cannot (discretely) pierce a rectangle containing the corner of the L-shape since then the edge qp would not be a Delaunay edge. Thus, in order to avoid edges that pierce other rectangles, it suffices to restrict q to lie between the upper and lower barriers at p. Thus Step 1 above ensures that edges found in Step 2 don't pierce any of the rectangles. Similarly, Step 3 ensures that the edges we add in Step 2 don't intersect previously added edges.

The overall time taken by the data structures used by Step 1 is $O((m + n) \log m)$ since it takes $O(\log m)$ time to insert or delete the key-value pair corresponding to any of the m rectangles, and it takes $O(\log m)$ time to query the data structure for the upper and lower barriers at any of the n points. The overall time taken by the data structure in Step 2 is $O(n \log^2 n)$ since there are most $O(n)$ insert, delete, and query operations, and the total number of points reported in all the queries together is $O(n)$. The overall time taken by the data structure in Step 3 is $O(n \log n)$ we only add $O(n)$ edges in the algorithm and the query corresponding to each edge takes $O(\log n)$ time. Each of the reported points is removed from the data structure but since each point is removed only once, the overall time for such removals is also $O(n \log n)$. The overall running time of our algorithm is therefore $O(n \log^2 n + (m + n) \log m)$.

References

1 Emmanuelle Anceaume, Maria Gradinariu, Ajoy Kumar Datta, Gwendal Simon, and Antonino Virgillito. A semantic overlay for self-peer-to-peer publish/subscribe. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 22–22. IEEE, 2006.

2 Daniel Antunes, Claire Mathieu, and Nabil H. Mustafa. Combinatorics of local search: An optimal 4-local Hall's theorem for planar graphs. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 8:1–8:13, 2017. doi:10.4230/LIPICS.ESA.2017.8.

3 Roberto Baldoni, Roberto Beraldi, Vivien Quema, Leonardo Querzoni, and Sara Tucci-Piergiovanni. Tera: topic-based event routing for peer-to-peer architectures. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, pages 2–13, 2007.

4 Roberto Baldoni, Roberto Beraldi, Leonardo Querzoni, and Antonino Virgillito. Efficient publish/subscribe through a self-organizing broker overlay and its application to SIENA. *The Computer Journal*, 50(4):444–459, 2007. doi:10.1093/COMJNL/BXM002.

5 Aniket Basu Roy, Sathish Govindarajan, Rajiv Raman, and Saurabh Ray. Packing and covering with non-piercing regions. *Discrete & Computational Geometry*, 2018.

6 Sergey Bereg, Krzysztof Fleszar, Philipp Kindermann, Sergey Pupyrev, Joachim Spoerhase, and Alexander Wolff. Colored non-crossing Euclidean Steiner forest. In *Algorithms and Computation: 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings*, pages 429–441. Springer, 2015. doi:10.1007/978-3-662-48971-0_37.

7 Sergey Bereg, Minghui Jiang, Boting Yang, and Binhai Zhu. On the red/blue spanning tree problem. *Theoretical computer science*, 412(23):2459–2467, 2011. doi:10.1016/J.TCS.2010.10.038.

8 Ulrik Brandes, Sabine Cornelsen, Barbara Pampel, and Arnaud Sallaberry. Blocks of hypergraphs: applied to hypergraphs and outerplanarity. In *Combinatorial Algorithms: 21st International Workshop, IWOCA 2010, London, UK, July 26-28, 2010, Revised Selected Papers 21*, pages 201–211. Springer, 2011. doi:10.1007/978-3-642-19222-7_21.

9 Ulrik Brandes, Sabine Cornelsen, Barbara Pampel, and Arnaud Sallaberry. Path-based supports for hypergraphs. *Journal of Discrete Algorithms*, 14:248–261, 2012. doi:10.1016/J.JDA.2011.12.009.

10 Gerth Stølting Brodal and Konstantinos Tsakalidis. Dynamic planar range maxima queries. In *International Colloquium on Automata, Languages, and Programming*, pages 256–267. Springer, 2011. doi:10.1007/978-3-642-22006-7_22.

11 Kevin Buchin, Marc J van Kreveld, Henk Meijer, Bettina Speckmann, and KAB Verbeek. On planar supports for hypergraphs. *Journal of Graph Algorithms and Applications*, 15(4):533–549, 2011. doi:10.7155/JGAA.00237.

12 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discret. Comput. Geom.*, 48(2):373–392, 2012. doi:10.1007/S00454-012-9417-5.

13 Raphaël Chand and Pascal Felber. Semantic peer-to-peer overlays for publish/subscribe networks. In *Euro-Par 2005 Parallel Processing: 11th International Euro-Par Conference, Lisbon, Portugal, August 30-September 2, 2005. Proceedings 11*, pages 1194–1204. Springer, 2005. doi:10.1007/11549468_130.

14 Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Constructing scalable overlays for pub-sub with many topics. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 109–118, 2007. doi:10.1145/1281100.1281118.

15 Vincent Cohen-Addad and Claire Mathieu. Effectiveness of local search for geometric optimization. In *Proceedings of the Thirty-first International Symposium on Computational Geometry*, SoCG '15, pages 329–343, Dagstuhl, Germany, 2015. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICS.SOCG.2015.329.

16 Ding-Zhu Du. An optimization problem on graphs. *Discrete applied mathematics*, 14(1):101–104, 1986. doi:10.1016/0166-218X(86)90010-7.

17 Ding-Zhu Du and Dean F Kelley. On complexity of subset interconnection designs. *Journal of Global Optimization*, 6(2):193–205, 1995. doi:10.1007/BF01096768.

18 Ding-Zhu Du and Zevi Miller. Matroids and subset interconnection design. *SIAM journal on discrete mathematics*, 1(4):416–424, 1988. doi:10.1137/0401042.

19 Frédéric Havet, Dorian Mazauric, Viet-Ha Nguyen, and Rémi Watrigant. Overlaying a hypergraph with a graph with bounded maximum degree. *Discrete Applied Mathematics*, 319:394–406, 2022. doi:10.1016/J.DAM.2022.05.022.

20 Jun Hosoda, Juraj Hromkovič, Taisuke Izumi, Hirotaka Ono, Monika Steinová, and Koichi Wada. On the approximability and hardness of minimum topic connected overlay and its special instances. *Theoretical Computer Science*, 429:144–154, 2012. doi:10.1016/J.TCS.2011.12.033.

21 Ferran Hurtado, Matias Korman, Marc van Kreveld, Maarten Löffler, Vera Sacristán, Akiyoshi Shioura, Rodrigo I Silveira, Bettina Speckmann, and Takeshi Tokuyama. Colored spanning graphs for set visualization. *Computational Geometry*, 68:262–276, 2018. doi:10.1016/J.COMGEO.2017.06.006.

22 David S Johnson and Henry O Pollak. Hypergraph planarity and the complexity of drawing Venn diagrams. *Journal of graph theory*, 11(3):309–325, 1987. doi:10.1002/JGT.3190110306.

23 Ephraim Korach and Michal Stern. The clustering matroid and the optimal clustering tree. *Mathematical Programming*, 98:385–414, 2003. doi:10.1007/S10107-003-0410-X.

24 Erik Krohn, Matt Gibson, Gaurav Kanade, and Kasturi Varadarajan. Guarding terrains via local search. *Journal of Computational Geometry*, 5(1):168–178, 2014. doi:10.20382/JOCG.V5I1A9.

25 Nabil H Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010. doi:10.1007/S00454-010-9285-9.

26 Melih Onus and Andréa W Richa. Minimum maximum-degree publish–subscribe overlay network design. *IEEE/ACM Transactions on Networking*, 19(5):1331–1343, 2011. doi:10.1109/TNET.2011.2144999.

27 Rajiv Raman and Saurabh Ray. Constructing planar support for non-piercing regions. *Discrete & Computational Geometry*, 64(3):1098–1122, 2020. doi:10.1007/S00454-020-00216-W.

28 Rajiv Raman and Saurabh Ray. On the geometric set multicover problem. *Discret. Comput. Geom.*, 68(2):566–591, 2022. doi:10.1007/s00454-022-00402-y.

29 Rajiv Raman and Karamjeet Singh. On hypergraph supports, 2024. arXiv:2303.16515.

30 AA Voloshina and VZ Feinberg. Planarity of hypergraphs. In *Doklady Akademii Nauk Belarusi*, volume 28, pages 309–311. Akademii Nauk Belarusi F Scorina Pr 66, room 403, Minsk, Byelarus 220072, 1984.

31 TRS Walsh. Hypermaps versus bipartite maps. *Journal of Combinatorial Theory, Series B*, 18(2):155–163, 1975.

32 Alexander Aleksandrovich Zykov. Hypergraphs. *Russian Mathematical Surveys*, 29(6):89, 1974.