On Minimum Generalized Manhattan Connections

Antonios Antoniadis[1], Margarita Capretto[2], Parinya Chalermsook[3], Christoph Damerius[4(✉)], Peter Kling[4], Lukas Nölke[5], Nidia Obscura Acosta[3], and Joachim Spoerhase[3]

[1] University of Twente, Enschede, The Netherlands
a.antoniadis@utwente.nl
[2] Universidad Nacional de Rosario, Rosario, Argentina
[3] Aalto University, Espoo, Finland
{parinya.chalermsook,nidia.obscuraacosta,joachim.spoerhase}@aalto.fi
[4] University of Hamburg, Hamburg, Germany
{christoph.damerius,peter.kling}@uni-hamburg.de
[5] University of Bremen, Bremen, Germany
noelke@uni-bremen.de

Abstract. We consider minimum-cardinality Manhattan connected sets with arbitrary demands: Given a collection of points P in the plane, together with a subset of pairs of points in P (which we call *demands*), find a minimum-cardinality superset of P such that every demand pair is connected by a path whose length is the ℓ_1-distance of the pair. This problem is a variant of three well-studied problems that have arisen in computational geometry, data structures, and network design: (i) It is a node-cost variant of the classical Manhattan network problem, (ii) it is an extension of the binary search tree problem to arbitrary demands, and (iii) it is a special case of the directed Steiner forest problem. Since the problem inherits basic structural properties from the context of binary search trees, an $O(\log\ n)$-approximation is trivial. We show that the problem is NP-hard and present an $O(\sqrt{\log n})$-approximation algorithm. Moreover, we provide an $O(\log\ \log\ n)$-approximation algorithm for complete k-partite demands as well as improved results for unit-disk demands and several generalizations. Our results crucially rely on a new lower bound on the optimal cost that could potentially be useful in the context of BSTs.

Keywords: Manhattan networks · Binary search tree · NP-hardness

1 Introduction

Given a collection of points $P \subset \mathbb{R}^2$ on the plane, the *Manhattan Graph* G_P of P is an undirected graph with vertex set $V(G_P) = P$ and arcs $E(G_P)$ that connect any vertically- or horizontally-aligned points. Point p is said to be *Manhattan-connected (M-connected)* to point q if G_P contains a shortest rectilinear path

The full version of this paper [1] can be found at https://arxiv.org/abs/2010.14338.

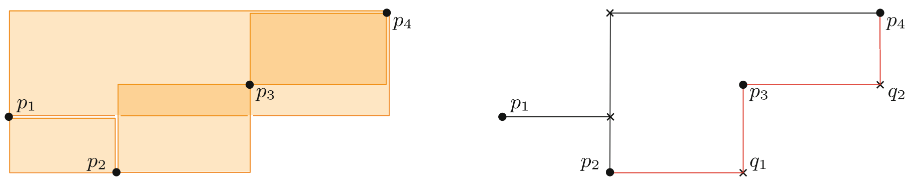


Fig. 1. Left: A Manhattan instance with input points in P drawn as black disks and demands in D drawn as orange rectangles. Right: The Manhattan Graph $G_{P \cup Q}$ of a feasible solution Q, with points in Q drawn as crosses. Points p_2 and p_4 are Manhattan-connected via the red path p_2 - q_1 - p_3 - q_2 - p_4. Points p_1 and p_3 are not Manhattan-connected but also not a demand pair in D. (Color figure online)

from p to q (i.e. a path of length $||p - q||_1$). In this paper, we initiate the study of the following problem: Given points $P \subset \mathbb{R}^2$ and demands $D \subseteq P \times P$, we want to find a smallest set $Q \subset \mathbb{R}^2$ such that every pair of vertices in D is M-connected in $G_{P \cup Q}$. We call this problem Minimum Generalized Manhattan Connections (MinGMConn), see Fig. 1 for an illustration. Variants of this problem have appeared and received a lot of attention in many areas of theoretical computer science, including data structures, approximation algorithms, and computational geometry. Below, we briefly discuss them, as well as the implications of our results in those contexts.

Binary Search Trees (BSTs). The *Dynamic Optimality Conjecture* [19] is one of the most fundamental open problems in dynamic data structures, postulating the existence of an $O(1)$-competitive binary search tree. Despite continuing efforts and important progress for several decades (see, e.g., [3,7,10,11,18] and references therein), the conjecture has so far remained elusive, with the best known competitive ratio of $O(\log \log n)$ obtained by Tango trees [11]. Even in the offline setting, the best known algorithm is also a $O(\log \log n)$-approximation; the problem is not even known to be NP-hard. Demaine, Harmon, Iacono, Kane, and Pătraşcu [10] showed that approximating BST is equivalent (up to a constant in the approximation factor) to approximating the node-cost Manhattan problem with "evolving demand" (that is, points added to the solution create demands to all existing points).[1]

The long-standing nature of the $O(\log \log n)$ upper bound could suggest the lower bound answer. However, the understanding of lower bound techniques for BSTs has been completely lacking: It is not even known whether the problem is NP-hard! Our work is inspired by the following question. *Is it NP-hard to (exactly) compute a minimum-cost binary search tree?* We are, unfortunately, unable to answer this question. In this paper, we instead present a proof that a natural generalization of the problem from the geometric point of view (which

[1] In fact, the problem stated in [10] is called MinASS which appears different from Manhattan problem, but they can be shown to be equivalent, see the Appendix of the full version [1].

is exactly our MinGMConn) is NP-hard[2]. We believe that our construction and its analysis could be useful in further study of the BST problem from the perspective of lower bounds.

Edge-Cost Manhattan Problem. Closely related to MinGMConn is the edge-cost variant of Manhattan Network [15]: Given $P \subset \mathbb{R}^2$, our goal is to compute $Q \subset \mathbb{R}^2$ such that every pair in P is M-connected in $G_{P \cup Q}$, while minimizing the total lengths of the edges used for the connections. The problem is motivated by various applications in city planning, network layouts, distributed algorithms and VLSI circuit design, and has received attention in the computational geometry community. Since the edge-cost variant is NP-hard [6], the focus has been on approximation algorithms. Several groups of researchers presented 2-approximation algorithms [5,16], and this has remained the best known approximation ratio. Generalizations of the edge-cost variant have been proposed and studied in two directions: In [9], the authors generalize the Manhattan problem to higher dimension $\mathbb{R}^d$ for $d \geqslant 2$. The arbitrary-demand case was suggested in [5]. An $O(\log\ n)$-approximation algorithm was presented in [8], which remains the best known ratio. Our MinGMConn problem can be seen as an analogue of [8] in the node-cost setting. We present an improved approximation ratio of $O(\sqrt{\log\ n})$, therefore, raising the possibility of similar improvements in the edge-cost variants.

Directed Steiner Forests (DSF). MinGMConn is a special case of *node-cost directed Steiner forest (DSF)*: Given a directed graph $G = (V, E)$ and pairs of terminals $\mathcal{D} \subseteq V \times V$, find a minimum cardinality subset $S \subseteq V$ such that $G[S]$ contains a path from s to t for all $(s, t) \in \mathcal{D}$. DSF is known to be highly intractable, with hardness $2^{\log^{1-\epsilon}|V|}$ unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{poly\log\ n})$ [12]. The best known approximation ratios are slightly sub-linear [4,13]. Manhattan problems can be thought of as natural, tractable special cases of DSF, with approximability between constant and logarithmic regimes. For more details, see [9].

1.1 Our Contributions

In this paper, we present both hardness and algorithmic results for MinGMConn.

Theorem 1. *The* MinGMConn *problem is NP-hard, even if no two points in the input are horizontally or vertically aligned.*

This result can be thought of as a first step towards developing structural understanding of Manhattan connectivity w.r.t. lower bounds. We believe such understanding would come in handy in future study of binary search trees in the geometric view.

Next, we present algorithmic results. Due to the BST structures, an $O(\log\ n)$-approximation is trivial. The main ingredient in obtaining a sub-logarithmic

[2] Demaine et al. [10] prove NP-hardness for MinGMConn with uniform demands but allow the input to contain multiple points on the same row. Their result is incomparable to ours.

approximation is an approximation algorithm for the case of "few" x-coordinates. Formally, we say an input instance is *s-thin* if points in P lie on at most s different x-coordinates.

Theorem 2. *There exists an efficient $O(\log\, s)$-approximation algorithm for s-thin instances of* MinGMConn.

In fact, our algorithm produces solutions with $O(\log\, s \cdot \mathsf{IS}(P, D))$ points, where $\mathsf{IS}(P, D) \leqslant \mathsf{OPT}(P, D)$ is the cardinality of a *boundary independent set* (a notion introduced below). This is tight up to a constant factor, as there is an input (P, D) on s different columns such that $\mathsf{OPT}(P, D) = \Omega(\mathsf{IS}(P, D) \log\, s)$; see the Appendix of [1].

This theorem, along with the boundary independent set analysis, turns out to be an important building block for our approximation result, which achieves an approximation ratio that is sublogarithmic in n.

Theorem 3. *There is an $O(\sqrt{\log\, n})$-approximation algorithm for* MinGMConn.

This improves over the trivial $O(\log\, n)$-approximation and may grant some new hope with regards to an improvement over this factor for the edge-cost variants.

We provide improved approximation ratios for several settings when the graph formed by the demands has a special structure. For example, we obtain an $O(\log \log\, n)$-approximation algorithm for MinGMConn when the demands form a complete k-partite graph, and an $O(1)$-approximation for unit-disk demands; see the Appendix of [1].

1.2 Overview of Techniques

The NP-hardness proof is based on a reduction to 3-SAT. In contrast to the uniform case of MinGMConn (where there is a demand for each two input points), the non-uniform case allows us to encode the structure of a 3-SAT formula in a geometrical manner: we can use demand rectangles to form certain "paths" (see Fig. 2). We exploit this observation in the reduction design by translating clauses and variables into *gadgets*, rectangular areas with specific placement of input points and demands (see Fig. 3). Variable gadgets are placed between clause gadgets and a dedicated *starting point.* The crux is to design the instance such that a natural solution to the intra- and inter-gadget demands connects the starting point to either the positive or the negative part of each variable gadget. And, the M-paths leaving a variable gadget from that part can all reach *only* clauses with a *positive* appearance or *only* clauses with a *negative* appearance of that variable respectively. We refer to such solutions as *boolean solutions*, as they naturally correspond to a variable assignment. Additional demands between the starting point and the clause gadgets are satisfied by a boolean solution if and only if it corresponds to a satisfying variable assignment. The main part of the proof is to show that any small-enough solution is a boolean solution.

In the study of any optimization (in particular, minimization) problem, one of the main difficulties is to come up with a strong lower bound on the cost of an optimal solution that can be leveraged by algorithms. For binary search trees, many such bounds were known, and the strongest known lower bound is called an *independent rectangle bound (IR).* However, IR is provably too weak for the purpose of MinGMConn, that is, the gap between the optimum and IR can be as large as $\Omega(n)$. We propose to use a new bound, which we call *vertically separable demands (VS).* This bound turns out to be relatively tight and plays an important role in both our hardness and algorithmic results. In the hardness result, we use our VS bound to argue about the cost of the optimum in the soundness case.

Our $O(\sqrt{\log n})$-approximation follows the high-level idea of [2], which presents a geometric $O(\log \log n)$-approximation for BST. Roughly speaking, it argues (implicitly) that two combinatorial properties, which we refer to as (A) and (B), are sufficient for the existence of an $O(\log \log n)$-approximation: (A) the lower bound function is "subadditive" with respect to a certain instance partitioning, and (B) the instance is "sparse" in the sense that for any input (P, D), there exists an equivalent input (P', D') such that $|P'| = O(\mathsf{OPT}(P, D))$. In the context of BST, (A) holds for the Wilber bound and (B) is almost trivial to show.

In the MinGMConn problem, we prove that Property (A) holds for the new VS bound. However, proving Property (B) seems to be very challenging. We instead show a corollary of Property (B): There is an $O(\log s)$-approximation algorithm for MinGMConn, where s is the number of columns containing at least one input point. The proof of this relaxed property is the main new ingredient of our algorithmic result and is stated in Theorem 2. Finally, we argue that this weaker property still suffices for an $O(\sqrt{\log n})$-approximation algorithm. For completeness, we discuss special cases where we prove that Property (B) holds and thus an $O(\log \log n)$-approximation exists. See the full version [1].

1.3 Outlook and Open Problems

Inspired by the study of structural properties of Manhattan connected sets and potential applications in BSTs, we initiate the study of MinGMConn by proving NP-hardness and giving several algorithmic results.

There are multiple interesting open problems. First, can we show that the BST problem is NP-hard? We hope that our construction and analysis using the new VS bound would be useful for this purpose. Another interesting open problem is to obtain a $o(\log n)$-approximation for the edge-cost variant of the generalized Manhattan network problem.

Finally, it can be shown that our VS bound is sandwiched between OPT and IR. It is an interesting question to study the tightness of the VS bound when estimating the value of an optimal solution. Is VS within a constant factor from the optimal cost of BST? Can we approximate the value of VS efficiently within a constant factor?

2 Model and Preliminaries

Let $P \subset \mathbb{R}^2$ be a set of points on the plane. We say that points $p, q \in P$ are *Manhattan-connected (M-connected)* in P if there is a sequence of points $p = x_0, x_1, \dots, x_k = q$ such that (i) the points x_i and x_{i+1} are horizontally or vertically aligned for $i = 0, \dots, k-1$, and (ii) the total length satisfies $\sum_{i=0}^{k-1} ||x_i - x_{i+1}||_1 = ||p - q||_1$.

In the *minimum generalized Manhattan connections* (MinGMConn) *problem*, we are given a set of *input points* P and their placement in a rectangular grid with integer coordinates such that there are no two points in the same row or in the same column. Additionally, we are given a set $D \subseteq \{(p,q)|p,q \in P\}$ of *demands*. The goal is to find a set of points Q of minimum cardinality such that p and q are M-connected with respect to $P \cup Q$ for all $(p,q) \in D$. Denote by $\mathsf{OPT}(P,D)$ the size of such a point set. We differentiate between the points of P and Q by calling them *input points* and *auxiliary points*, respectively. Since being M-connected is a symmetrical relation, we typically assume $x(p) < x(q)$ for all $(p,q) \in D$. Here, $x(p)$ and $y(p)$ denote the x- and y-coordinate of a point p, respectively. In our analysis, we sometimes use the notations $[n] \coloneqq \{1, 2, \dots, n\}$ and $[n]_0 \coloneqq [n] \cup \{0\}$, where $n \in \mathbb{N}$.

Connection to Binary Search Trees. In the *uniform* case, i.e. $D = \{(p,q)|p,q \in P\}$, this problem is intimately connected to the Binary Search Tree (BST) problem in the geometric model [10]. Here, we are given a point set P and the goal is to compute a minimum set Q such that *every pair* in $P \cup Q$ is M-connected in $P \cup Q$. Denote by $\mathsf{BST}(P)$ the optimal value of the BST problem.

Independent Rectangles and Vertically Separable Demands. Following Demaine et al. [10], we define the independent rectangle number which is a lower bound on $\mathsf{OPT}(P,D)$. For a demand $(p,q) \in D$, denote by $R(p,q)$ the (unique) axis-aligned closed rectangle that has p and q as two of its corners. We call it the *demand rectangle* corresponding to (p,q). Two rectangles $R(p,q), R(p',q')$ are called *non-conflicting* if none contains a corner of the other in its interior. We say a subset of demands $D' \subseteq D$ is *independent*, if all pairs of rectangles in D' are non-conflicting. Denote by $\mathsf{IR}(P,D)$ the maximum integer k such that there is an independent subset D' of size k. We refer to k as the *independent rectangle number.*

For uniform demands, the problem admits a 2-approximation. Here, the independent rectangle number plays a crucial role. Specifically, it was argued in Harmon's PhD thesis [17] that a natural greedy algorithm costs at most the independent rectangle number and thus yields a 2-approximation. In our generalized demand case, however, the independent rectangle number turns out to be a bad estimate on the value of an optimal solution. Instead, we consider the notion of vertically separable demands, used implicitly in [10].

We say that a subset of demands $D' \subseteq D$ is *vertically separable* if there exists an ordering $R_1, R_2, \dots, R_k$ of its demand rectangles and vertical line segments $\ell_1, \ell_2 \dots, \ell_k$ such that ℓ_i connects the respective interiors of top and bottom boundaries of R_i and does not intersect any R_j, for $j > i$. For an input (P, D), denote by $\mathsf{VS}(P, D)$ the maximum cardinality of such a subset. We call a set of demands D *monotone*, if either $y(p) < y(q)$ for all $(p, q) \in D$ or $y(p) > y(q)$ for all $(p, q) \in D$. We assume the former case holds as both are symmetrical. In the following, we argue that VS is indeed a lower bound on OPT (the proof can be found in the full version [1]).

Lemma 4 [10]. *Let (P, D) be an input for* MinGMConn. *If D is monotone, then $\mathsf{IR}(P, D) \leqslant \mathsf{VS}(P, D) \leqslant \mathsf{OPT}(P, D)$. In general $\frac{1}{2}\mathsf{IR}(P, D) \leqslant \mathsf{VS}(P, D) \leqslant 2 \cdot \mathsf{OPT}(P, D)$.*

The charging scheme described Lemma 4's proof injectively maps a demand rectangle R to a point of the optimal solution in R. This implies the following corollary.

Corollary 5. *Let D be a vertically separable, monotone set of demands and Q a feasible solution. If $|Q| = |D|$, there is a bijection $c\colon Q \to D$ such that $q \in R(c(q))$ for all $q \in Q$. In particular, for $Q' \subseteq Q$ there are at least $|Q'|$ demands from D that each covers some $q \in Q'$.*

In general, the independent rectangle number and the maximum size of a vertically separable set are incomparable. By Lemma 4, $\mathsf{IR}(P, D) \leqslant 2 \cdot \mathsf{VS}(P, D)$. However, $\mathsf{IR}(P, D)$ may be smaller than $\mathsf{VS}(P, D)$ up to a factor of n. To see this, consider n diagonally shifted copies of a demand, e.g. $R_i = R\big((i, i), (i+n, i+n)\big)$, for $i = 1, \dots, n$. Here, $\mathsf{IR}(P, D) = 1$ and $\mathsf{VS}(P, D) = n$. Thus, the concept of vertical separability is more useful as a lower bound.

3 NP-Hardness

In this section, we show Theorem 1 by reducing the 3-SAT problem to MinGMConn. In 3-SAT, we are given a formula ϕ consisting of m clauses $C_1, C_2, \dots, C_m$ over n variables $X_1, X_2, \dots, X_n$, each clause consisting of three literals. The goal is to decide whether ϕ is satisfiable. For our reduction, we construct a MinGMConn instance (P_ϕ, D_ϕ) and a positive integer $\alpha = \alpha(\phi)$ such that (P_ϕ, D_ϕ) has an optimal solution of size α if and only if ϕ is satisfiable (see the full version [1]). This immediately implies Theorem 1. In the following, we identify a demand $d \in D_\phi$ with its demand rectangle $R(d)$. This allows us to speak, for example, of intersections of demands, corners of demands, or points covered by demands.

Our construction of the MinGMConn instance (P_ϕ, D_ϕ) is based on different *gadgets* and their connections among each other. A gadget can be thought of as a rectangle in the Euclidean plane that contains a specific set of input points and demands between these. In the following, we give a coarse overview of our construction, describing how gadgets are placed and how they interact (Fig. 2). Moreover, we try to convey the majority of the intuition behind our reduction. Because of space constraints the actual proof of the NP-hardness is given in the full version [1].

Overview of the Construction. For each clause C_j, we create a *clause gadget* GC_j and for each variable X_i, a *variable gadget* GX_i. Clause gadgets are arranged along a descending diagonal line, so all of GC_j is to the bottom-right of GC_{j-1}. Variable gadgets are arranged in the same manner. This avoids unwanted interference among different clause and variable gadgets, respectively. The variable gadgets are placed to the bottom-left of all clause gadgets.

For each positive occurrence of a variable X_i in a clause C_j, we place a dedicated *connection point* $p_{ij}^+ \in P_\phi$ as well as suitable *connection demands* from p_{ij}^+ to a dedicated inner point of GX_i and to a dedicated inner point of GC_j. Their purpose is to force optimal MinGMConn solutions to create specific M-paths (going first up and then right in a narrow corridor) connecting a variable to the clauses in which it appears positively. We call the area covered by these two demands a *(positive) variable-clause path.* Similarly, there are connection points $p_{ij}^- \in P_\phi$ with suitable demands for negative appearances of X_i in C_j, creating a *(negative) variable-clause path* (going first right and then up in a narrow corridor).

Finally, there is a *starting point* $S \in P_\phi$ to the bottom-left of all other points. It has a demand to a *clause point* c_j in the top-right of each clause gadget GC_j (an *SC demand*) and to a *variable point* x_i in the bottom-left of each variable gadget GX_i (an *SX demand*). The inside of clause gadgets simply provides different entrance points for the variable-clause paths, while the inside of variable gadgets forces an optimal solution to choose between using either only positive or only negative variable-clause paths. We will use these choices inside variable gadgets to identify an optimal solution for (P_ϕ, D_ϕ) with a variable assignment for ϕ.

The *clause gadget* GC_j for clause C_j contains the *clause point* c_j and three *(clause) literal points* $\ell_{j1}, \ell_{j2}, \ell_{j3}$. The clause point is in the top-right. The literal points represent the literals of C_j and form a descending diagonal within the gadget such that positive are above negative literals. For each literal point ℓ_{jk}, there is a demand (ℓ_{jk}, c_j). Moreover, if ℓ_{jk} is positive and corresponds to the variable X_i, then there is a *(positive) connection demand* (p_{ij}^+, ℓ_{jk}). Similarly, if ℓ_{jk} is negative, there is a *(negative) connection demand* (p_{ij}^-, ℓ_{jk}). Finally, there is the *SC demand* (S, c_j).

The *variable gadget* GX_i for variable X_i contains the *variable point* x_i, two *(variable) literal points* x_i^+, x_i^-, one *demand point* d_i, as well as n_i^+ positive and n_i^- negative *literal connectors* x_{ik}^+ and x_{ik}^-, respectively. Here, n_i^+ and n_i^- are from $[m]_0$ and denote the number of positive and negative occurrences of X_i in ϕ, respectively. The variable point is in the bottom-left. The literal connectors and the demand point form a descending diagonal in the top-right, with the positive literal connectors above and the negative literal connectors below the demand point. The literal points x_i^+, x_i^- lie in the interior of the rectangle spanned by x_i and d_i, close to the top-left and bottom-right corner respectively. They are moved slightly inward to avoid identical x- or y-coordinates. Inside the gadgets, we have demands of the form (x_i^+, x_{ik}^+) and (x_i^-, x_{ik}^-) between literal points and literal connectors, (x_i^+, d_i) and (x_i^-, d_i) between literal points and the demand point, as well as (x_i, d_i) between the variable point and the demand point (an *XD demand*). Towards the outside, we have the *positive/negative connection demands*between literal points and literal connectors (x^+_{ik}, p^+_{ij}) if the k-th positive literal of X_i occurs in C_j and (x^-_{ik}, p^-_{ij}) if the k-th negative literal of X_i occurs in C_j as well as the *SX demand* (S, x_i).

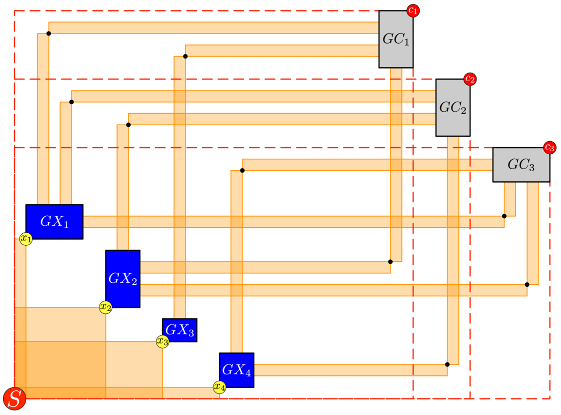


Fig. 2. MinGMConn instance (P_ϕ, D_ϕ) for $\phi = (X_1 \vee \neg X_2 \vee X_3) \wedge (X_1 \vee X_2 \vee \neg X_4) \wedge (\neg X_1 \vee \neg X_2 \vee X_4)$. Input points are shown as (red, yellow, or black) disks. For clause and variable gadgets, we show only the clause points c_j and the variable points x_i; their remaining inner points and demands are illustrated in Fig. 3. The small black disks represent the connection points p^+_{ij}, p^-_{ij}. Non-SC demands are shown as shaded, orange rectangles, while SC demands are shown as dashed, red rectangles. (Color figure online)

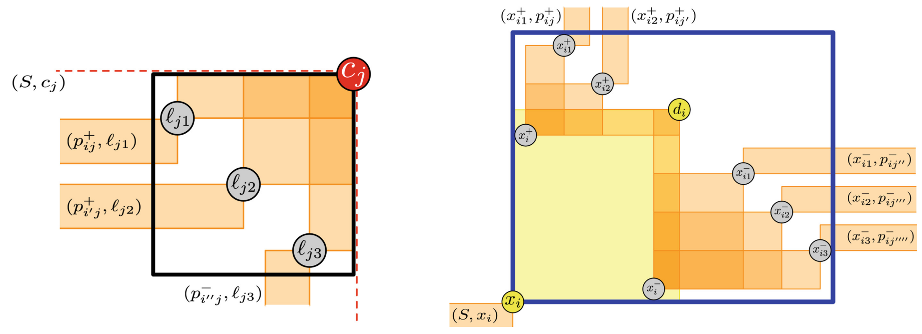
Intuition of the Reduction. Our construction is such that non-SC demands (including those within gadgets) form a monotone, vertically separable demand set. Thus, for

$$D_{\overline{SC}} := \{d \in D_\phi | d \text{ is not an } SC \text{ demand}\} \quad \text{and} \quad \alpha = \alpha(\phi) := |D_{\overline{SC}}|, \qquad (1)$$

Lemma 4 implies that any solution Q_ϕ for (P_ϕ, D_ϕ) has size at least α.

The first part of the reduction shows that if ϕ is satisfiable, then there is an (optimal) solution Q_ϕ of size α. This is proven by constructing a family of *boolean solutions*. These are (partial) solutions Q_ϕ that can be identified with a variable assignment for ϕ and that have the following properties: Q_ϕ has size α and satisfies all non-SC demands. Additionally, it can satisfy an SC demand (S, c_j) only by going through some variable x_i, where such a path exists if and only if C_j is satisfied by the value assigned to X_i by (the variable assignment) Q_ϕ. In particular, if ϕ is satisfiable, there is a boolean solution Q_ϕ satisfying all SC demands. This implies that Q_ϕ is a solution to (P_ϕ, D_ϕ) of (optimal) size α.

We then provide the other direction of the reduction, stating that if there is a solution Q_ϕ for (P_ϕ, D_ϕ) of size α, then ϕ is satisfiable. Its proof is more



(a) Clause gadget GC_j for the clause $C_j = X_i \vee X_{i'} \vee \neg X_{i''}$.

(b) Variable gadget GX_i for X_i occurring twice positively and thrice negatively.

Fig. 3. Examples for a clause and a variable gadgets. As in Fig. 2, input points are shown as circles and SC demands are shown as dashed, red rectangles. The XD demand (x_i, d_i) is shown as a shaded, yellow rectangle. All remaining (non-SC and non-XD) demands are again shown as shaded, orange rectangles. (Color figure online)

involved and is made possible by careful placement of gadgets, connection points, and demands. (See the full version [1] for the complete proof.) In a first step, we show that the small size of Q_ϕ implies that different parts of our construction each must be satisfied by only a few, dedicated points from Q_ϕ. For example, Q_ϕ has to use exactly n points to satisfy the n SX demands (S, x_i). Another result about "triangular" instances (e.g., the triangular grid formed by the n SX demands, see Fig. 2) states that, here, optimal solutions must lie on grid lines inside the "triangle". See the full version [1]. We conclude that any M-path from S to a clause point c_j must go through exactly one variable point x_i. Similarly, we show that the $6m$ connection demands (forming the $3m$ variable-clause paths) are satisfied by $6m$ points from Q_ϕ and, since they are so few, each of these points lies in the corner of a connection demand. This ensures that M-paths cannot cheat by, e.g., "jumping" between different variable-clause paths. More precisely, such a path can be entered only at the variable gadget where it starts and be left only at the clause gadget where it ends.

All that remains to show is that there cannot be two M-paths entering a variable gadget GX_i (which they must do via x_i) such that one leaves through a positive and the other through a negative variable-clause path. We can then interpret Q_ϕ as a boolean solution (the variable assignment for X_i being determined by whether M-paths leave GX_i through positive or through negative variable-clause paths). Since Q_ϕ satisfies all demands, in particular all SC demands, the corresponding variable assignment satisfies all clauses.

4 An Approximation Algorithm for s-Thin Instances

In this section, we present an approximation algorithm for s-thin instances (points in P lie on at most s distinct x-coordinates). In particular, we allow more than one point to share the same x-coordinate. However, we still require any two points to have distinct y-coordinates. We show an approximation ratio of $O(\log s)$, proving Theorem 2.

An *x-group* is a maximal subset of P having the same x-coordinate. Note that an $O(\log s)$-approximation for s-thin instances can be obtained via a natural "vertical" divide-and-conquer algorithm that recursively divides the s many x-groups in two subinstances with roughly $s/2$ many x-groups each. (Section 5 considered a more general version of this, subdividing into an arbitrary number of subinstances.) The analysis of this algorithm uses the number of input points as a lower bound on OPT. However, such a bound is not sufficient for our purpose of deriving an $O(\sqrt{\log n})$-approximation.

In this section, we present a different algorithm, based on "horizontal" divide-and-conquer (after a pre-processing step to sparsify the set of y-coordinates in the input via minimum hitting sets). Using horizontal rather than vertical divide-and-conquer may seem counter-intuitive at first glance as the number of y-coordinates in the input is generally unbounded in s. Interestingly enough, we can give a *stronger* guarantee for this algorithm by bounding the cost of the approximate solution against what we call a *boundary independent set.* Additionally, we show that the size of a such set is always upper bounded by the maximum number of vertically separable demands. This directly implies Theorem 2, since $2\,\mathsf{OPT}$ is an upper bound on the number of vertically separable demands (c.f. Lemma 4). Even more importantly, our stronger bound allows us to prove Theorem 3 in the next section since vertically separable demands fulfill the subadditivity property mentioned in the introduction. In the proof of Theorem 3, an arbitrary $O(\log s)$-approximation algorithm would not suffice.

By losing a factor 2 in the approximation ratio, we may assume that the demands are monotone (we can handle pairs with $x(p) < x(q)$ and $y(p) > y(q)$ symmetrically).

Definition 6 (Left & right demand segments). *Let (P, D) be an input instance. For each $R(p, q) \in Q$, denote by $\lambda(p, q)$ the vertical segment that connects $(x(p), y(p))$ and $(x(p), y(q))$. Similarly, denote by $\rho(p, q)$ the vertical segment that connects $(x(q), y(p))$ and $(x(q), y(q))$. That is, $\lambda(p, q)$ and $\rho(p, q)$ are simply the left and right boundaries of rectangle $R(p, q)$.*

Boundary Independent Sets. A *left boundary independent set* consists of pairwise non-overlapping segments $\lambda(p, q)$, a *right boundary independent set* of pairwise non-overlapping segments $\rho(p, q)$. A *boundary independent set* refers to either a left or a right boundary independent set. Denote by $\mathsf{IS}(P, D)$ the size of a maximum boundary independent set.

The following lemma implies that it suffices to work with boundary independent sets instead of vertical separability. The main advantage of doing so, is that (i) for IS, we do not have to identify any ordering of the demand subset, (ii) one can compute $\mathsf{IS}(P, D)$ efficiently, and (iii) we can exploit geometric properties of interval graphs, as we will do below.

```
Algorithm 1: HorizontalDC($P, D, \mathcal{R}$)
input  : Instance $(P, D)$ with rows $\mathcal{R}$
output : Feasible solution to $(P, D)$ computed via horizontal
         divide-and-conquer
1 $Q \leftarrow \emptyset$; $m \leftarrow$ median of $\mathcal{R}$;
2 $D_m \leftarrow \{ (p, q) \in D \mid y(p) \leqslant m \leqslant y(q) \}$;
3 foreach $(p, q) \in D_m$ do
4     $Q \leftarrow Q \cup \{(x(p), m), (x(q), m)\}$;
5 $D_t \leftarrow \{ (p, q) \in D \mid y(p) > m \}$;    $D_b \leftarrow \{ (p, q) \in D \mid y(q) < m \}$;
6 $P_t \leftarrow \{ p, q \mid (p, q) \in D_t \}$;            $P_b \leftarrow \{ p, q \mid (p, q) \in D_b \}$;
7 $\mathcal{R}_t \leftarrow \{ r \in \mathcal{R} \mid r > m \}$;              $\mathcal{R}_b \leftarrow \{ r \in \mathcal{R} \mid r < m \}$;
8 $Q \leftarrow Q \cup$ HorizontalDC$(P_t, D_t, \mathcal{R}_t) \cup$ HorizontalDC$(P_b, D_b, \mathcal{R}_b)$;
9 return $Q$;
```

Lemma 7. *For any instance (P, D) we have that $\mathsf{IS}(P, D) \leqslant \mathsf{VS}(P, D)$. Moreover, one can compute a maximum boundary independent set in polynomial time.*

Algorithm Description. Our algorithm, which we call HorizontalManhattan, produces a Manhattan solution of cost $O(\log\, s) \cdot \mathsf{IS}(P, D)$, where s is the number of x-groups in P. The algorithm initially computes a set of "crucial rows" $\mathcal{R} \subseteq \mathbb{R}$ by computing a minimum hitting set in the interval set $\mathcal{I} = \{[y(p), y(q)] \mid (p, q) \in D\}$. In particular, the set $\mathcal{R}$ has the following property. For each $j \in \mathcal{R}$, let ℓ_j be a horizontal line drawn at y-coordinate j. Then the lines $\{\ell_j\}_{j \in \mathcal{R}}$ stab every rectangle in $\{R(p, q)\}_{(p,q) \in D}$. The following observation follows from the fact that the interval hitting set is equal to the maximum interval independent set.

Observation. $|\mathcal{R}| \leqslant \mathsf{IS}(P, D)$.

After computing $\mathcal{R}$, the algorithm calls a subroutine HorizontalDC (see Algorithm 1), which recursively adds points to each such row in a way that guarantees a feasible solution. We now proceed to the analysis of the algorithm.

Lemma 8 (Feasibility). *The algorithm* HorizontalManhattan *produces a feasible solution in polynomial time.*

Lemma 9 (Cost). *For any s-thin instance (P, D), algorithm* HorizontalManhattan *outputs a solution of cost $O(\log\, s) \cdot \mathsf{IS}(P, D)$.*

Proof. Let $r = |\mathcal{R}|$ be the number of rows computed in HorizontalManhattan. Define $L = \{\lambda(p, q) \mid [y(p), y(q)] \in I\}$ to be the set of corresponding left sides of the demands in I. In particular, the segments in L are disjoint and $|L| = r$. We upper bound the cost of our solution as follows. For each added point, define a *witness interval*, witnessing its cost. The total number of points is then roughly bounded by the number of witness intervals, which we show to be $O(\log\, s)\mathsf{IS}(P, D)$.

We enumerate the recursion levels of Algorithm 1 from 1 to $\lceil \log r \rceil$ in a top-down fashion in the recursion tree. In each recursive call, at most s many points are added to Q in line 4—one for each distinct x-coordinate. Hence, during the first $\lceil \log r \rceil - \lceil \log s \rceil$ recursion levels at most $s \cdot 2^{\lceil \log r \rceil - \lceil \log s \rceil} = O(s \cdot \frac{r}{s}) = O(r)$ many points are added to Q in total. We associate each of these points with one unique left side in L in an arbitrary manner. For each of these points, we call its associated left side the witness of this point.

For any point added to Q in line 4 in one of the last $\lceil \log s \rceil$ recursion levels, pick the first (left or right) side of a demand rectangle that led to including this point. More precisely, if, in line 4, we add point $(x(p), m)$ to Q for the first time (which means that this point has not yet been added to Q via a different demand) then associate $\lambda(p, q)$ as a witness. Analogously, if we add $(x(q), m)$ for the first time then associate $\rho(p, q)$ as a witness.

Overall, we have associated to each point in the final solution a uniquely determined witness, which is a left or a right side of some demand rectangle. Note that any (left or right) side of a rectangle may be assigned as a witness to two solution points (once in the top recursion levels and once in the bottom levels). In such a case we create a duplicate of the respective side and consider them to be distinct witnesses.

Two witnesses added in the last $\lceil \log s \rceil$ recursion levels can intersect only if the recursive calls lie on the same root-to-leaf path in the recursion tree. Otherwise, they are separated by the median row of the lowest common ancestor in the recursion tree and cannot intersect. With this observation and the fact that the witnesses in L form an independent set, we can bound the maximum clique size in the intersection graph of all witnesses by $1 + \lceil \log s \rceil$.

This graph is an interval graph. Since interval graphs are perfect [14], there exists a $(1 + \lceil \log s \rceil)$-coloring in this graph. Hence, there exists an independent set of witnesses of size $1/(\lceil \log s \rceil + 1)$ times the size of the Manhattan solution. Taking all left or all right sides of demands in this independent set (whichever is larger) gives a *boundary* independent set of size at least $1/(2(1 + \lceil \log s \rceil))$ times the cost of the Manhattan solution. □

We conclude the section by noting that the proof of Theorem 2 directly follows by combining Lemmata 4, 7 and 9. As mentioned in the introduction, the factor $O(\log\ s)$ in Lemma 9 is tight in the strong sense that there is a MinGMConn instance (P, D) with s distinct x-coordinates such that $\mathsf{OPT}(P, D) = \Omega(\mathsf{IS}(P, D) \log s)$. See the appendix of the full version [1].

5 A Sublogarithmic Approximation Algorithm

In this section, we give an overview of how to leverage the $O(\log s)$-approximation for s-thin instances to design an $O(\sqrt{\log n})$-approximation algorithm for general instances.

Sub-instances. Let (P, D) be an instance of MinGMConn and let B be a bounding box for P, that is, $P \subseteq B$. Let $\mathcal{S} = \{S_1, \ldots, S_s\}$ be a collection of

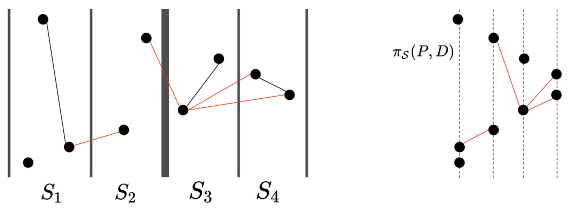


Fig. 4. An illustration of the inter-strip instance. Each strip S_i is collapsed into one column. The red demands are demands between pairs of points lying inside different strips. The black demands are demands that are handled by intra-strip instances. (Color figure online)

s vertical strips, ordered from left to right, that are obtained by drawing $s-1$ vertical lines that partition B. We naturally create $s+1$ sub-instances as follows. (See also Fig. 4.) First, we have s *intra-strip instances* $\{(P_i, D_i)\}_{i\in[s]}$ such that $P_i = P \cap S_i$ and $D_i = D \cap (S_i \times S_i)$. Next, we have the *inter-strip instance* $\pi_{\mathcal{S}}(D) = (P', D')$ where P' is obtained by collapsing each strip in $\mathcal{S}$ into a single column and D' is obtained from collapsing demands accordingly. For each point $p \in P$, denote by $\pi_{\mathcal{S}}(p)$ a copy of p in P' after collapsing. Note that this is a simplified description of the instances that avoids some technicalities. For a precise definition, see the appendix in the full version [1].

Sub-additivity of VS. The following is our sub-additivity property that we use crucially in our divide-and-conquer algorithm.

Lemma 10. *If* (P, D) *is an instance of* MinGMConn *with strip subdivision* $\mathcal{S}$*, then*

$$\mathsf{VS}(P,D) \geqslant \mathsf{VS}(\pi_{\mathcal{S}}(P,D)) + \sum_{S\in\mathcal{S}} \mathsf{VS}(P\cap S, D\cap(S\times S)).$$

Divide-and-Conquer. Choose the strips $\mathcal{S}$ so that $s = |\mathcal{S}| = 2^{\sqrt{\log n}}$. Thus the inter-strip instance admits an approximation of ratio $O(\log s) = O(\sqrt{\log n})$; in fact, we obtain a solution of cost $O(\sqrt{\log n})\mathsf{VS}(\pi_{\mathcal{S}}(P,D))$. We recursively solve each intra-strip instance (P_i, D_i), and combine the solutions from these $s+1$ sub-instances. (Details on how the solution can be combined are deferred to [1].)

We show by induction on the number of points that for any instance (P, D) the cost of the computed solution is $O(\sqrt{\log n})\mathsf{VS}(P,D)$. (Here, we do not take into account the cost incurred by combining the solutions to the sub-instances.) By induction hypothesis, we have for each (P_i, D_i) a solution of cost $O(\sqrt{\log n})\mathsf{VS}(P_i, D_i)$ since $|P_i| < |P|$. Note that we cannot use the induction hypothesis for the inter-strip instance since $|P'| = |P|$, which is why we need the $O(\log s)$-approximation algorithm. Using sub-additivity we obtain:

$$O(\sqrt{\log n})\Big(\mathsf{VS}(\pi_{\mathcal{S}}(P,D)) + \sum_{S\in\mathcal{S}} \mathsf{VS}(P\cap S, D\cap(S\times S))\Big) = O(\sqrt{\log n})\mathsf{VS}(P,D).$$

There is an additional cost incurred by combining the solutions of the sub-instances to a feasible solution of the current instance. In the full version [1], we argue that this can be done at a cost of $O(\mathsf{OPT})$ for each of the $\log n/\log s = \sqrt{\log n}$ many levels of the recursion. (This prevents us from further improving the approximation factor by picking $s = 2^{o(\sqrt{\log n})}$.)

References

1. Antoniadis, A., et al.: On minimum generalized Manhattan connections. CoRR abs/2010.14338 (2020). https://arxiv.org/abs/2010.14338
2. Chalermsook, P., Chuzhoy, J., Saranurak, T.: Pinning down the strong Wilber 1 bound for binary search trees. arXiv preprint arXiv:1912.02900 (2019)
3. Chalermsook, P., Goswami, M., Kozma, L., Mehlhorn, K., Saranurak, T.: Pattern-avoiding access in binary search trees. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, pp. 410–423. IEEE (2015)
4. Chekuri, C., Even, G., Gupta, A., Segev, D.: Set connectivity problems in undirected graphs and the directed Steiner network problem. ACM Trans. Algorithms (TALG) **7**(2), 1–17 (2011)
5. Chepoi, V., Nouioua, K., Vaxes, Y.: A rounding algorithm for approximating minimum Manhattan networks. Theoret. Comput. Sci. **390**(1), 56–69 (2008)
6. Chin, F.Y., Guo, Z., Sun, H.: Minimum Manhattan network is NP-complete. Discrete Comput. Geom. **45**(4), 701–722 (2011)
7. Cole, R.: On the dynamic finger conjecture for splay trees. part II: the proof. SIAM J. Comput. **30**(1), 44–85 (2000)
8. Das, A., Fleszar, K., Kobourov, S., Spoerhase, J., Veeramoni, S., Wolff, A.: Approximating the generalized minimum Manhattan network problem. Algorithmica **80**(4), 1170–1190 (2018)
9. Das, A., Gansner, E.R., Kaufmann, M., Kobourov, S., Spoerhase, J., Wolff, A.: Approximating minimum Manhattan networks in higher dimensions. Algorithmica **71**(1), 36–52 (2015)
10. Demaine, E.D., Harmon, D., Iacono, J., Kane, D., Pătraşcu, M.: The geometry of binary search trees. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 496–505. SIAM (2009)
11. Demaine, E.D., Harmon, D., Iacono, J., Pătraşcu, M.: Dynamic optimality-almost. SIAM J. Comput. **37**(1), 240–251 (2007)
12. Dodis, Y., Khanna, S.: Design networks with bounded pairwise distance. In: Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing, pp. 750–759 (1999)
13. Feldman, M., Kortsarz, G., Nutov, Z.: Improved approximation algorithms for directed Steiner forest. J. Comput. Syst. Sci. **78**(1), 279–292 (2012)
14. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Annals of Discrete Mathematics, vol. 57. North-Holland Publishing Co., NLD (2004)
15. Gudmundsson, J., Levcopoulos, C., Narasimhan, G.: Approximating a minimum Manhattan network. Nordic J. Comput. **8**(2), 219–232 (2001). http://dl.acm.org/citation.cfm?id=766533.766536
16. Guo, Z., Sun, H., Zhu, H.: A fast 2-approximation algorithm for the minimum Manhattan network problem. In: Fleischer, R., Xu, J. (eds.) AAIM 2008. LNCS, vol. 5034, pp. 212–223. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68880-8_21

17. Harmon, D.D.K.: New bounds on optimal binary search trees. Ph.D. thesis, Massachusetts Institute of Technology (2006)
18. Iacono, J., Langerman, S.: Weighted dynamic finger in binary search trees. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 672–691. SIAM (2016)
19. Sleator, D.D., Tarjan, R.E.: Self-adjusting binary search trees. J. ACM (JACM) **32**(3), 652–686 (1985)