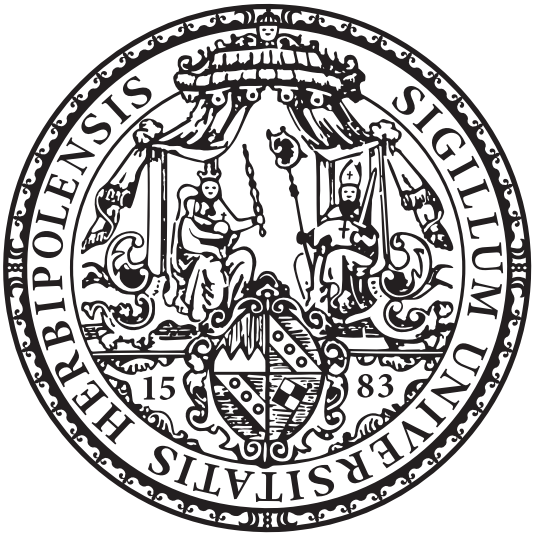
Bachelor Thesis

Variations of the Generalized Minimum Manhattan Network Problem

Tim Gerlach

Date of Submission: 14.02.2022
Advisors: Dr. habil. Joachim Spoerhase
Prof. Dr. Alexander Wolff



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen und Komplexität

Abstract

Given a set of points $T \subseteq \mathbb{R}^2$ as well as a set of point pairs $C \subseteq \binom{T}{2}$, the Generalized Minimum Manhattan Network problem is the problem of computing a minimum-length set of axis-aligned line segments that connect the pairs of points from C with a shortest path according to the Manhattan (L_1) metric. We show a constant factor approximation for instances where the bounding boxes of point pairs all contain certain points or one of a grid of points. We also introduce a new related problem and give a Mixed Integer Programming statement for it.

Zusammenfassung

Wir betrachten das „Generalized Minimum Manhattan Network“ Problem: Gegeben eine Punktmenge $T \subseteq \mathbb{R}^2$ und eine Menge von Punktpaaren $C \subseteq \binom{T}{2}$, berechne eine Menge minimaler Gesamtlänge von vertikalen und horizontalen Strecken, die alle Punktpaare aus C mit einem kürzesten Pfad unter der Manhattan- oder L_1-Metrik verbindet. Für mehrere Teilprobleme wird ein Konstantfaktorapproximationsalgorithmus gezeigt. Außerdem wird ein neues, verwandtes Problem definiert und als ganzzahliges lineares Programm formuliert.

Contents

1 Introduction

Network design is among the most fundamental areas in combinatorial optimization. It deals with finding a subgraph of a given input graph satisfying certain restrictions, such as connectivity, and optimizing some objective function, such as total cost of included edges. Well-known examples include the minimum spanning tree problem, the Steiner tree problem, or the traveling salesman problem.

For example, consider the Steiner tree problem:

Definition 1 (Steiner tree problem)**.** *Given a set of* terminals T*, and a set of* nonterminals U*, that form the graph* $G = (V \cup U, E)$ *with* $E = \binom{T \cup U}{2}$*, as well as a metric* $c : E \to \mathbb{R}^+$*, find a subgraph that connects all vertices from* T*, while minimizing the total cost of included edges.*

The resulting subgraph, or *network*, for this problem is called a *Steiner tree*. Note that optimal solutions are actually always trees, as cycles would add redundant connections.

Geometric network design refers to computing a set of geometric paths connecting a given point set. A geometric variant of the Steiner tree problem is the Euclidean Steiner tree problem, where $T \subseteq \mathbb{R}^2$ is a point set in the Euclidean plane, all other points in $\mathbb{R}^2$ are nonterminals, and c is the Euclidean distance between any point pair.

Geometrically connecting a set of points has relevance even in non-obvious applications like VLSI circuit design [GLN01, Maß15], and there are several well-studied problems asking to do this under certain restrictions and with differing optimization goals.

Another geometric variant of the Steiner tree problem is on the space $\mathbb{R}^2$ with the Manhattan-metric L_1. This metric is defined as follows:

Definition 2 (L_1 metric)**.** *Let* $p = (p_1, p_2, \ldots, p_d), q = (q_1, q_2, \ldots, q_d)$ *be two points in* $\mathbb{R}^d$ *for* $d \geq 1$*. Then, the Mahattan metric for those points is defined as:*

$$L_1(p, q) := \sum_{i=1}^{d} |p_i - q_i|$$

In $\mathbb{R}^2$, this makes the notion of distance between two points correspond to projecting a point pair to both x- and y-axis, and then adding both Euclidean distances between the respective projected points. A line segment is *axis-aligned*, if it is either strictly vertical or strictly horizontal. A path or a network is *rectilinear*, it it consists only of axis-aligned line segments. Computing a Steiner tree on this space using the Manhattan metric means finding a minimum-length rectilinear network.

While in Euclidean spaces the shortest connection between any two points is a straight line, in L_1 spaces the shortest connection between two points is an L_1*-shortest path*, sometimes also called a Manhattan- or M-path.

Definition 3 (L_1-shortest path)**.** *An L_1-shortest path between two points p and q is a sequence of axis-aligned line segments connecting p and q, whose total length is equal to $L_1(p, q)$.*

In Euclidean spaces, a connection of shortest length between any two given points is some unique straight line, but in L_1 spaces, there are generally an infinite number of shortest connections.

This leads to a problem related to the Steiner tree problem where we add the constraint of containing an L_1-shortest path between each input point pair. This problem is called the *Minimum Manhattan Network problem* (MMN).

Definition 4 (Minimum Manhattan Network Problem)**.** *Let $T \subseteq \mathbb{R}^2$ be a set of designated points of the plane, called* terminal points*. A* Manhattan Network *N is a set of rectilinear line segments connecting the points from T, such that for any pair of points $\{u, v\} \in \binom{T}{2}$, there is a path in N of length $L_1(u, v)$. The Minimum Manhattan Network Problem is the problem of constructing such a network with minimal total line length.*

As opposed to Steiner trees, optimal solutions for MMN may contain cycles. More generally, there is also the problem of constructing a *t-spanner*, which is a network connecting each pair of points using at most t times the distance of a shortest connection [GNS16]. In Euclidean spaces, a 1-spanner is just the set of all lines between all pairs of points. In L_1 spaces, computing a 1-spanner is exactly equal to the Minimum Manhattan Network problem.

1.1 The Generalized Minimum Manhattan Network Problem

While MMN is already an interesting problem, one can generalize it, by relaxing the definition from connecting all terminals to just requiring a connection between a given set of terminal pairs. This problem is known as the *Generalized Minimum Manhattan Network Problem* (GMMN).

Definition 5 (Generalized Minimum Manhattan Network Problem)**.** *Given a set of points $T \subseteq \mathbb{R}^2$ and a set of unordered pairs of points $C \subseteq \binom{T}{2}$, The Generalized Minimum Manhattan Network Problem is the problem of finding a set of rectilinear line segments of minimal total length, such that any point pair $\{u, v\} \in C$ is connected via a path of line segments, whose length is equal to the Manhattan-distance between u and v.*

This problem happens to not only generalize MMN, but also the *Rectilinear Steiner Arborescence Problem* (RSA).

Definition 6 (Rectilinear Steiner Arborescence)**.** *Given a set T of points contained in the first quadrant of $\mathbb{R}^2$, find a an acyclic set of rectilinear line segments such that every point of T is connected to the origin by an L_1-shortest path.*

In this problem, connections are not between any pair of input points, but rather between each input point and the origin.

1.2 Rectangle Stabbing

Another relevant geometric problem is called *Rectangle Stabbing*. While not a network design problem, it is a variant of geometric set cover which proved to be useful at approaching GMMN in the past [DFK+18].

Definition 7 ((Rectangle) Stabbing)**.** *Given a set of axis-aligned rectangles R in $\mathbb{R}^2$, find a set of horizontal line segments with minimal total length, such that for each rectangle $r \in R$ there is at least one line segments which intersects both the left and right boundary of r.*

1.3 Related Work

Minimum Manhattan Networks

MMN was first brought up by Gudmundsson et al. [GLN01]. They also found efficient factor-8 and factor-4 approximation algorithms for it. Since then, it has received quite some attention. The approximation ratio was first improved to 3 by Benkert, Wolff, Widmann and Shirabe [BWWS06], then to 2 by Chepoi, Nouioua and Vax [CNV08]. A claim for a factor-1.5 approximation by Seibert and Unger [SU05] was refuted by Fuchs and Schulze [FS08]. Fuchs and Schulze also gave another factor-3 approximation algorithm. It was unclear during all this time whether MMN was actually $\mathcal{NP}$-hard, which was settled first for the three-dimensional version by Muñoz, Seibert and Unger [MSU09], where they also proved that it is $\mathcal{NP}$-hard to approximate 3D-MMN within a factor of $(1 + 10^{-}5)$. The two-dimensional case was shown to be strongly $\mathcal{NP}$-hard to solve by Chin, Guo and Sun [CGS11]. For higher dimensional cases, Das et al. gave an $\mathcal{O}(n^{\varepsilon})$ approximation for $\varepsilon > 0$ [DGK+15].

It is still unknown whether the two-dimensional MMN allows a polynomial time approximation scheme (PTAS), that is, a polynomial time factor-$(1 + \varepsilon)$ approximation algorithm for arbitrarily small $\varepsilon > 0$.

Generalized Minimum Manhattan Networks

GMMN was introduced by Chepoi et al. [CNV08] who gave the 2-approxmation for MMN. Including MMN as a subproblem, GMMN is also known to be $\mathcal{NP}$-hard. Das et al. [DFK+18] showed that GMMN in d dimensions can be approximated up to a factor of $\mathcal{O}(\log^{d+1} n)$, as well as $\mathcal{O}(\log n)$ for the two-dimensional case. Another $\mathcal{O}(\log n)$ approximation was given by Funke and Seybold [FS14] with a stronger bound if it is known that the input has a limited so-called *scale-diversity*. There are dynamic programming approaches for instances with intersection graphs of bounded treewidth and degree, discovered by Schnizler [Sch15], and intersection graphs that are trees by Masumura et al. [MOY21].

There is no general approximation known which is better than $\mathcal{O}(\log n)$. It is still open whether there are constant-factor approximations or approximation schemes.

Rectilinear Steiner Arborescence

Rectilinear Steiner Arborescence was proven to be $\mathcal{NP}$-hard by Shi and Su [SS05], but to also admit constant factor approximation algorithms. For example, Ramnath gave a factor-2 approximation [Ram03]. There is also a PTAS for the 2-dimensional case, which was shown by Zachariasen [Zac00].

Rectangle Stabbing

The rectangle stabbing problem was introduced by Chan et al. [CvDF+18], who showed it to be $\mathcal{NP}$-hard, as well as giving a constant factor approximation. As a special geometric set cover problem, a $\mathcal{O}(\log n)$ approximation follows directly.

Eisenbrand et al. [EGSV21] further investigated the problem and came up with both a simple constant factor approximation as well as a quasi-polynomial time approximation scheme. The most recent development is the discovery of a PTAS by Khan et al. [KSW21].

1.4 Contribution

There are multiple interesting subproblems of GMMN, which have good approximations and are generally well-understood. The most important question regarding the generalized version is whether there is still a constant factor approximation or even a PTAS or QPTAS. In this thesis, we make some small progress on this question by giving $O(1)$ approximations for various special cases, as well as defining a new special case that may give some more insight into the problem. These cases were encountered as obstacles in a naïve $\mathcal{APX}$ hardness proof attempt, which implies that they should be approachable, and are natural problems in their own right.

First, we show that the family of instances where all rectangles contain a common point can be efficiently approximated by employing known algorithms for RSA. Using this as a subroutine, we can also efficiently approximate all instances where all rectangles contain at least one grid point of some constant-sized grid. This, in turn, allows approximating all instances which only have similar sized rectangles.

In particular, we show Theorem 22, which states that given an instance P with a line grid with a spacing of γ and size $w \times h$, where all bounding boxes of terminal pairs contain at least one grid point, the solution for P can be approximated up to a factor of 126 in time $\mathcal{O}^*(2^{wh\gamma^{-2}})$.

We use this to show Theorem 24, which gives a factor 166 approximation for instances where the shortest bounding box side length and the largest bounding box side length are only different by a constant factor.

Finally, we introduce Monotone Stabbing as an intermediate problem between GMMN and Rectangle Stabbing, give a MIP formulation, and show that it has a large integrality gap.

1.5 Definitions

When talking about GMMN, it often makes sense to identify terminal pairs with their bounding rectangles:

Definition 8. *The* bounding box *of a terminal pair p, q is the closed rectangle spanned by the corner points p and q.*

A L_1-shortest path will always be contained within the bounding box between its end points.

Definition 9. *A terminal pair p, q* containing *a point x means that x lies within the bounding box of the pair.*

Definition 10. *Two terminal pairs* overlap*, if and only if either pair's bounding box contains a point of the other.*

Definition 11. *An instance is called* connected*, if and only if the graph that contains a vertex for each terminal pair and an edge between any two overlapping pairs are connected.*

An optimal GMMN solution for an instance P is denoted as N_P, its cost as $\mathsf{OPT}(P)$, or just N and OPT if there is only one relevant instance.

1.6 Motivation

While GMMN is NP-hard, solving it exactly by brute force is still an option for small instances. However, since GMMN is a geometric problem, the solution space is by nature of uncountably infinite size, which is not feasible to iterate. Since there are a lot of problems that are subproblems of GMMN, they all are also solved by brute-forcing GMMN, and the technique to restrict the solution space to something that is approachable is also applicable for all of them.

This technique was first brought up in the context of rectilinear Steiner trees, and the idea is to restrict the set of solutions to those that only contain line segments from the Hanan-Grid [Han66].

The Hanan-Grid is an axis-aligned grid, which has vertices on all x- and y-coordinates, on which the instance has a terminal point, as shown in Figure 1.1.

Given an instance P for GMMN, as well as some solution N_P, it is possible to show that there is another solution N'_P, with at most the same cost, which lies entirely on the Hanan-grid [FS14]. Since all terminal points lie on vertices of the Hanan-grid, this means that there are only a finite amount of line segments, which are either fully contained or fully excluded from N'_P.

Lemma 12. *The number of the line segments of the Hanan-Grid is bounded by $2n^2$, where n is the number of input terminal points.*

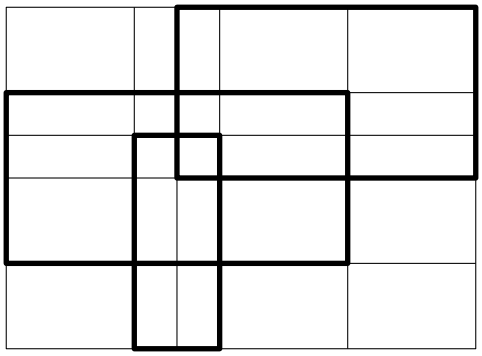


Fig. 1.1: An example of the Hanan-grid (thin lines) for a set of rectangles (thick lines)

Proof. Each vertex of the Hanan grid shares an x- or y-coordinate with an input terminal point, so there are at most n^2 vertices. Each vertex is incident to at most four grid lines, and each grid line is incident to exactly two vertices. Therefore, there are at most $n^2 \cdot \frac{4}{2}$ grid lines. □

This implies a simple brute-force algorithm which solves the problem by enumerating all 2^{n^2} relevant subsets of the Hanan-grid line segments, filters for the ones that give a valid solution for the input, and then returns the one with the lowest overall length. This runs in $\mathcal{O}^*(2^{n^2})$ time, since checking for valid solutions and potentially updating the minimum is simple to do in polynomial time.

This is a very inefficient algorithm, but it does find a perfect solution. Since this is often not a hard requirement, and there is not much hope for an algorithm that runs efficiently on large instances anyway, a lot of the work on the Generalized Minimum Manhattan Network Problem has gone into finding approximation algorithms. Still, even the best algorithms have non-constant approximation ratio.

This might suggest that GMMN does not admit constant-factor approximations, or at least no approximation schemes. There is a concept for this, which is called $\mathcal{APX}$-hardness. Informally, if a problem is $\mathcal{APX}$-hard, and $\mathcal{P} \neq \mathcal{NP}$, then there is no PTAS for it.

Definition 13 ($\mathcal{APX}$-hardness)**.** *$\mathcal{APX}$ is the set of all problems that allow a constant-factor, polynomial time approximation algorithm. A problem is* $\mathcal{APX}$-hard*, if there is a PTAS-reduction from all problems in $\mathcal{APX}$ to it.*

It is not known whether GMMN is $\mathcal{APX}$-hard or not. The following is a basic idea for a wrong $\mathcal{APX}$-hardness proof for GMMN, together with an explanation of the obstacles of this construction. These obstacles in turn lead to the special cases considered.

The proof idea is by reducing from a SAT variant used in the paper of Muñoz et al. [MSU09], where they showed $\mathcal{APX}$-hardness for three-dimensional MMN.

Definition 14 (SAT)**.** *Given a logical formula F in conjunctive normal form, find a variable assignment that satisfies F.*

Definition 15 (MAX-(3,B2)-SAT)**.** *Given a logical formula F in conjunctive normal form, where every clause contains exactly three variables, and every variable occurs exactly twice positive and twice negative, find the maximum number of clauses that can be satisfied by any variable assignment.*

In the paper, they proved a useful theorem, which is quoted here:

Theorem 16. *There exists a family of MAX-(3,B2)-SAT instances, containing* $1016n$ *clauses for some* n*, such that it is* $\mathcal{NP}$*-hard to distinguish between instances where* $(1016n - \varepsilon)$ *clauses can be satisfied and those where at most* $(1015 + \varepsilon)n$ *clauses can be satisfied (for arbitrary small* $\varepsilon < \frac{1}{2}$*).*

By embedding this problem within a GMMN instance, we can show $\mathcal{APX}$-hardness. The more clauses are satisfyable, the cheaper an optimal solution will have to become. Let c_1 be the cost of a solution satisfying $(1016 - \varepsilon)n$ clauses. Let c_2 be the minimum cost satisfying at most $(1015 + \varepsilon)n$ clauses. By construction, c_1 will have to be smaller than c_2. The goal is to have $c_2/c_1 > d > 1$ for a constant d. If this is the case, it would be $\mathcal{NP}$-hard to decide whether a solution falls into the former or the latter case, according to above theorem, but any PTAS could decide this efficiently by choosing an approximation factor less than d:

Let s be the cost of the solution returned by the PTAS. If $s \leq c_2$ holds, then $(1016-\varepsilon)n$ clauses must be satisfiable. On the other hand, if s is larger than c_2, the optimal solution is already larger than c_1. This can be obtained by dividing out the approximation factor: $s \leq d \cdot \mathsf{OPT}$, and $s > c_2$, so $\mathsf{OPT} \geq s/d > c_2/d > c_1$. Since even the optimal solution is not as cheap as a solution satisfying $(1015 + \varepsilon)n$ clauses, it is not possible to satisfy that many clauses.

Being able to decide this $\mathcal{NP}$-hard problem with a PTAS implies the $\mathcal{APX}$-hardness of the problem.

So, the idea is to construct some gadgets that together model a MAX-(3,B2)-SAT instance. All the gadgets presented will work correctly, but when combining them, the only way to stack them in two dimensions increases the optimal solution length drastically, making the gap between an optimal solution and an approximation vanish.

As a start, there needs to be some way to encode the variables. This is accomplished by adding rectangles, for which any optimal solution, and any solution sufficiently close to optimal, will contain exactly one of two sides. As shown in Figure 1.2, the gadget consists of two input point pairs arranged at the corners of a rectangle. $\{A, A'\}$ and $\{B, B'\}$ need to be connected. Let a be the distance between A and B and l be the distance between A and B'. If the paths do not overlap at all, then the total cost for connecting this gadget is $2a+2l$. If they overlap, the cost can be reduced down to $2a+l$.

Because all L_1-shortest paths are monotone, the path from A to A' will never increase in y-coordinate while the x-coordinate increases, and the path from B to B' will never decrease in y-coordinate. This means that the paths cannot meet, then split up, and then meet again. Therefore, if one can enforce that the paths overlap in the regions close to the terminal points defining the rectangle, they will overlap throughout the

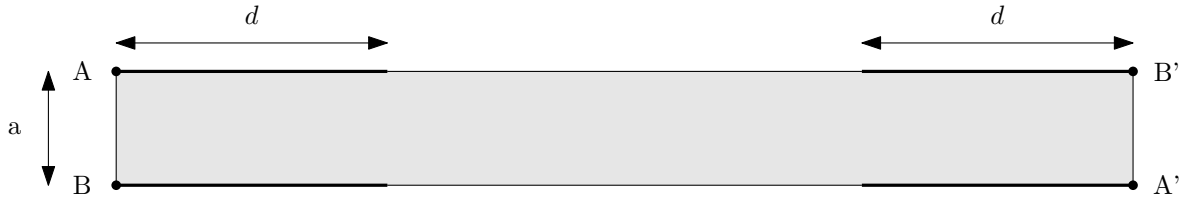


Fig. 1.2: A variable gadget

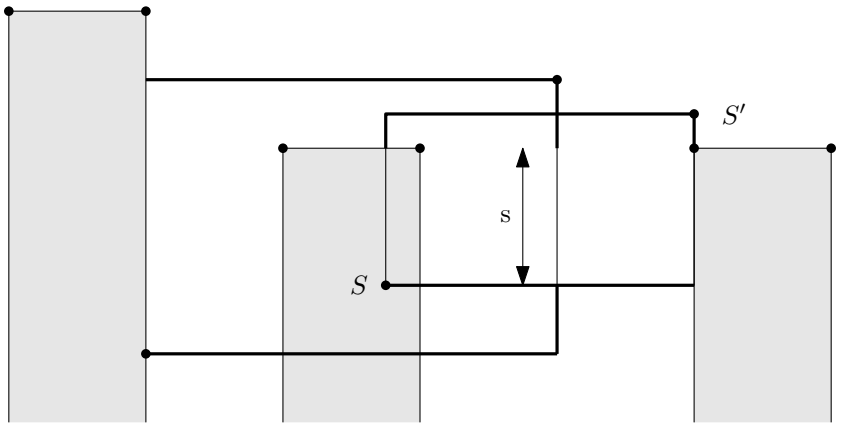


Fig. 1.3: A clause gadget

entire box. If one can additionally enforce that the overlapping paths lie on either the top or bottom boundary of the rectangle, this already is a sufficient variable gadget, encoding an assigment of 1 as the solution containing a line at the top, and a 0 as the solution containing a line at the bottom.

To guarantee these properties, further connections are added: If an input pair shares a common x- or y-coordinate, then the unique shortest Manhattan path between them is a straight line. Using this, one can ensure that any valid solution contains some line segments. These are drawn as bold lines in the image, and are used to enforce the needed property: If the variable line lies either at the top or bottom boundary, then it can share cost with one of the fixed lines, going through the center will increase the cost by at least d.

To encode the clauses, another gadget is defined, which sits at the end of three variable gadgets. If any of the variable gadgets is true, then it will be cheap to solve the rest of the gadget, but if none is, there will be some added cost. In particular, the path from S to S' incurs no additional cost, if either the variable gadgets at the sides have a line running through the $\{S, S'\}$ bounding box, or if the larger rectangle has a line at the right side.

Lastly, to ensure that a variable can occur in multiple clauses, and that each clause can be reached by any variable, we need both a *crossing* and a *splitter*. In both cases, variable gadgets run through each other, but in the first case, they are fully independent, while for the splitter there is an additional cost if both variable gadgets do not have the

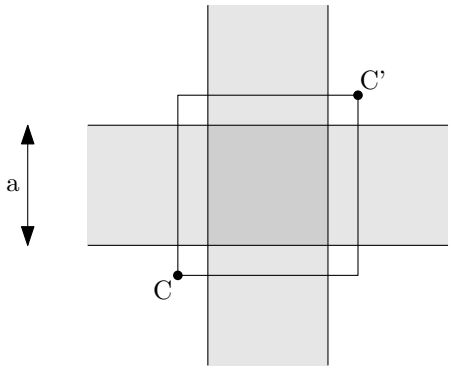


Fig. 1.4: A crossing gadget

same value. The splitter can also force both variable gadgets to have different values, which can be used to implement negative literals. For a crossing, both variable gadgets can simply through each other. Since any line orthogonal to the main direction of the gadget requires the main connections to split, no line can be shared between both gadgets without drastically increasing the total line length. For the splitter, another terminal point pair is added as in Figure 1.4. If the connecting lines of the variable gadgets form either the upper left or the lower right corner of the bounding box rectangle of the new pair, then connecting it is cheap. If they form one of the other corners, then they exclude each other from being useful in a solution, which adds at least a cost of a.

Inverting variables is necessary to encode negative occurences, but this can simply be done by mirroring the added terminal point pair at a splitter, which will change which pair of variable assignments are cheap and which are not.

When putting these gadgets together, each variable gadget must be able to reach every clause gadget. An arrangement may be to simply run all variable gadgets parallel, and branch out the currently required variables to the top using splitters as shown in Figure 1.5. The problem with this variable gadget grid is that it is too big: Let n be the number of variables in the input formula. There are in total $4n$ variable occurrences, by problem definition. In the worst case, each variable will have to run through almost the entire variable stack, with a minimum solution cost of $\mathcal{O}(an)$, or $\mathcal{O}(an^2)$ in total, where a is the width of a variable gadget. Since a can not become arbitrarily small without breaking the functionality of the crossing gadgets, any optimal solution will have cost quadratic in the number of input variables. With this, one cannot follow the argument of Muñoz et al., since the added cost of failing to satisfy a clause vanishes against the total solution cost.

While this construction therefore can not show $\mathcal{APX}$-hardness, it can hint at simpler substructures that, even if GMMN turns out to be $\mathcal{APX}$-hard, may still allow PTAS. In particular, the construction exhibits multiple interesting properties: All bounding box rectangles can be hit by an evenly spaced point grid, requiring only few points, it features rectangles that are very tall, but thin, and all of its rectangles are similarily sized. In the rest of this work, several possibly more approachable subproblems related

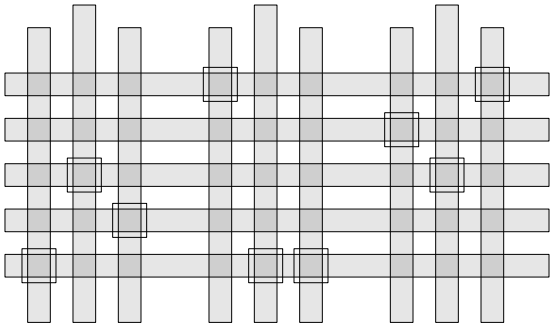


Fig. 1.5: A possible arrangement of variable gadgets

to these properties will be presented and discussed.

2 Common-Point GMMN

The first subproblem is a variant where all the bounding boxes of input pairs contain a common point, which is in following without loss of generality the origin O.

A *monotone instance* of GMMN is an instance, where for each input point pair the point with the larger x-coordinate also has the larger y-coordinate. The *Common-Point GMMN Variant*, abbreviated here as PGMMN, consists of solving only monotone instances to the GMMN Problem, where all bounding boxes of input point pairs contain the origin.

The rectilinear Steiner arborescence problem mentioned in the introduction can be phrased more directly as a GMMN subproblem by simply requiring all input pairs to contain the origin. Intuitively, to solve RSA, one has to connect some point set to the origin using only L_1-shortest paths with minimum total length. There is a known polynomial time approximation scheme for RSA which efficiently solves this problem up to a factor of $(1+\varepsilon)$, for arbitrary $\varepsilon > 0$ [Zac00].

It can be shown that this PTAS can also be employed to approximate PGMMN: First, every input to PGMMN can be transformed into an input for RSA by replacing each point pair $\{p, q\}$ with the pairs $\{p, O\}$ and $\{O, q\}$. In the original definition, there is a requirement for all points to lie in a common quadrant, but this restriction can be lifted: Since q needs to be strictly to the top right of p, and O lies within the bounding box of p and q, q will always be within the first quadrant, while p will lie in the third. Since the first and third quadrant don't share any points except O, and relevant L_1-shortest paths will be fully contained within them, it is possible to approximate RSA for the first and third quadrant separately and then merge the results.

Now, let ε be an arbitrary positive real number for the approximation factor. Let $\mathsf{OPT}_{\mathrm{PGMMN}}(I)$ be the cost of an optimal PGMMN solution of some instance I. Similarly, let $\mathsf{OPT}_{\mathrm{RSA}}(I)$ be the cost of an optimal RSA solution on the instance obtained by transforming I into an RSA instance. With c_I we denote the cost of a solution obtained by running the $(1+\varepsilon)$ approximation algorithm for RSA on I.

In order to show that PGMMN can also be approximated by $(1+\varepsilon)$, we need to show that the optimal solution for RSA on a transformed instance is at most as expensive as the optimal solution for PGMMN, and that any RSA solution can be transformed into an PGMMN solution with the same cost, since then we can obtain an c_I-cost solution for PGMMN for any I, and we have

$$c_I \leq (1+\varepsilon)\mathsf{OPT}_{\mathrm{RSA}}(I) \leq (1+\varepsilon)\mathsf{OPT}_{\mathrm{PGMMN}}(I)$$

Any RSA solution is in fact already a valid PGMMN solution: As above, input points to PGMMN can only lie in the first and third quadrant, since otherwise either O would not be contained in the bounding box or the instance would not be monotone. There is

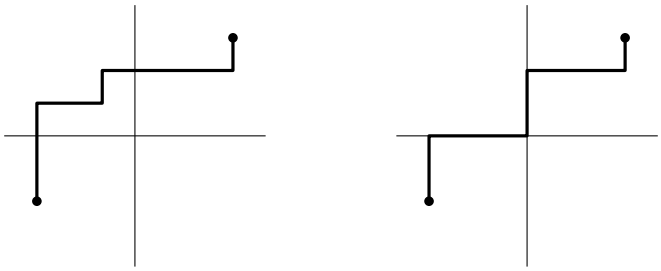


Fig. 2.1: A path between terminals before and after transformation

a L_1-shortest path from each terminal point in the third quadrant to O, and there is a path from O to every terminal in the first quadrant as well. L_1-shortest paths are always monotone. Since every point in a path in the first quadrant is strictly to the top left of the paths in the third quadrant, the union of a path from the first and a path from the third quadrant is monotone as well. Since any monotone rectilinear path between two points has the same length, the union of corresponding paths is also a shortest path. Therefore, the RSA solution contains a L_1-shortest path between any terminal in the first and any terminal in the third quadrant. All terminal pairs in the PGMMN input have to be distributed this way to contain O in their bounding box. All terminal pairs are connected in the RSA solution.

Showing that the optimal RSA solution is at most as expensive as the optimal PGMMN solution is done by a similar transformation. Let L be the set of line segments that solves some PGMMN instance I optimally. The claim is that there always is a set of line segments L' with the same total length, which solve RSA on I.

Consider X, the set of points that lie both on a line segment and on an axis. Since there are no terminals in quadrants two and four, any path through those quadrants can be replaced by the unique L_1-shortest path that contains both the points that were both on the path and in X, and the origin, as shown in Figure 2.1. After the replacement, a new set of line segments L' is obtained.

Observation 17. *If l is an L_1-shortest path from p to q, which goes through points u and v, then replacing the segment from u to v with any other L_1-shortest path from u to v is still a valid L_1-shortest path overall*

This is still a valid PGMMN solution. Potential overlapping connections in the original solution still overlap after the transformation, the transformation can never increase the cost of the solution. It is left to show that L' also solves the equivalent RSA instance. This can be seen by considering that every path connecting a terminal pair in the original input has to cross either the origin or at least two axes. In any case, after the replacements, every path will cross the origin, resulting in L_1-shortest paths between each terminal point and O. This yields a PTAS for PGMMN.

Theorem 18. *There is a factor $(1+\varepsilon)$ approximation algorithm for PGMMN for arbitrarily small $\varepsilon > 0$.*

A non-monotone instance can be $(2+\varepsilon)$-approximated by splitting it into two separate monotone instances, and approximating them individually.

Theorem 19. *There is a factor* $(2+\varepsilon)$ *approximation algorithm for non-monotone PGMMN for arbitrarily small* $\varepsilon > 0$.

3 Grid Approach

Extending the ideas of the previous GMMN Variant, combined with the brute-force approach, directly allows approximating a range of inputs that contain similarly-sized rectangles.

Grid-Solutions

Observation 20. *For solving any GMMN instance P, it is sufficient to individually consider its maximal connected sub-instances.*

This is the case since the feasible solutions for independent sub-instances can not interact with each other without intersecting bounding boxes.

Containing two grid points

Lemma 21. *Let G be a grid with a line spacing of γ, and side lengths w and h. If, for some GMMN-Instance P, all bounding boxes of terminal pairs contain at least two grid points, then the solution for P can be approximated with factor* 136 *in time $\mathcal{O}^*(2^{wh\gamma^{-2}})$.*

To this end, all terminals are snapped to the grid, where an optimal solution can be determined via brute force. The original terminals are then connected to the grid.

For each input terminal pair, choose a pair of snapped points on the grid by picking the grid points that are both closest to the respective terminal, and contained within the bounding box of the terminal pair. The snapped points always exist, since there are at least two points within each bounding rectangle. Additionally, they are always distinct between two corresponding terminals, since being closest to the same contained point would imply that there is only a single point within the rectangle.

Doing this for each terminal point pair generates a GMMN-Instance with at most four times the original number of terminals, that completely lies on the grid. Optimal solutions for GMMN can be transformed to lie on the Hanan-Grid induced by the input points, so optimal solutions for snapped instances generated this way can be transformed to lie on the grid G. Call this solution N_{GRID}.

Using brute force, N_{GRID} can be computed in $\mathcal{O}^*(2^{wh\gamma^{-2}})$: Since the entire solution lies on the grid, each grid segment can only either be fully contained in the solution, or not at all. All subsets of the $2 \cdot w/\gamma \cdot h/\gamma$ grid line segments can be enumerated in the stated time, and each subset can be checked for being a valid network in polynomial time relative to the grid size, since there are at most as many terminals as there are grid points.

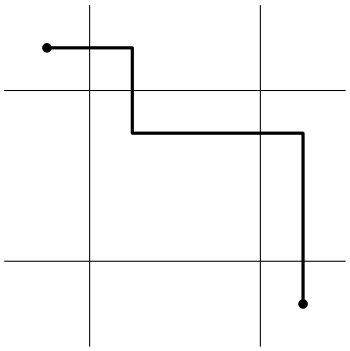


Fig. 3.1: Example path through grid cells

Finally, each terminal point needs to be connected to its corresponding snapped counterpart, which can be done by 2-approximating RSA in each grid cell, where the origin is the snapped point and all other input points are the original terminals.

This leads to a valid solution, since solving the snapped instance optimally is valid itself, and the extensions never violate any monotonicity, so the overall connections remain L_1-shortest paths.

To obtain the approximation ratio, let N_{A} be the solution obtained by this algorithm, with total cost A, and N and OPT the overall optimal solution and optimal cost for the input GMMN instance. With GRID denoting the cost of N_{GRID}, there is $\mathsf{GRID} \leq 8\mathsf{OPT}$: Connect all pairs using only grid line segments that are adjacent to a grid cell containing at least $\gamma/2$ units of the connection from N. Since there are only four grid line segments for each of these cells, this increases cost by at most a factor of $4\gamma/\frac{\gamma}{2}$, yielding a *snapped* solution, with cost at most $8\mathsf{OPT}$. And since GRID is the cost for an optimal solution on the grid, it has to be at least as good.

The snapped solutions are valid, since along the path from OPT, cells with $\gamma/2$ or more units are always at least diagonally adjacent: If it leaves any grid cell from the opposite side the line entered it, it will lead to the grid cell always containing at least γ line length. Only if the cell is left via a cell side that is adjacent to the side from which the cell was entered, the line through the cell may be shorter than $\gamma/2$. Assume without loss of generality that a cell was entered from the left and is exited to the bottom. This is illustrated in Figure 3.1. If it spent less than $\gamma/2$ line length in this cell, it will end up in the upper left quadrant of the next cell. From there, it needs to exit to the bottom or right, since all paths need to be monotone, and this makes up a line of length at least $\gamma/2$.

Now, consider without loss of generality only terminal points from terminal point pairs that lie at the top left corner of the bounding box of the pair. Observe that in the optimal solution, for each terminal point, a connection has to leave the grid cell to the right or to the bottom. Introduce line segments of length γ along the right and bottom boundaries of the grid cell. Call these additional lines the *boundary lines* of their respective grid cell. The original connection together with a boundary line connects the terminal points to their corresponding snapped terminal. These connections are L_1-shortest paths, since

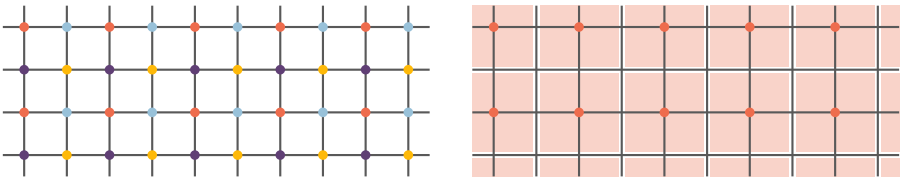


Fig. 3.2: Independent sub-grids, each color contains only independent instances

the connections from an optimal solution have to be, and the extension leads strictly down- and rightwards from the lowest, rightmost point of the optimal solution.

This forms a superset of an RSA solution, so by 2-approximating RSA in each grid cell, and aggregating the results, a set of connections are found that cost at most 2γ per cell plus $2\mathsf{OPT}$ overall.

The costs of the boundary lines added can also be bounded by a multiple of OPT: Since each bounding box contains at least two grid points, in N_{GRID}, there is at least one line leaving the grid cell, with a length of γ. The boundary lines can thus be said to be at most twice as expensive as an existing line in N_{GRID}, and this charge against a grid line can occur at most four times. This allows estimating the entire boundary line cost as less than $8\mathsf{GRID} \leq 64\mathsf{OPT}$.

The entire upper bound for A is thus $8\mathsf{OPT}$ for the grid solution, $64\mathsf{OPT}$ for the RSA approximations, and $64\mathsf{OPT}$ for the added boundary lines, which makes up a total cost of at most $136\mathsf{OPT}$.

Containing one grid point

Theorem 22. *Let G, γ, w, h, P be defined as above, except that the bounding boxes in P are allowed to contain only a single grid point. For such instances, there is a factor 152 approximation algorithm which runs in $\mathcal{O}^*(2^{wh\gamma^{-2}})$ time.*

These preconditions allow for a slightly worse constant-factor approximation, by first connecting all terminal pairs with only a single grid point independently of all other pairs, and then using the procedure from Lemma 21 on the remaining instance.

Approximating the solution for all terminal pairs containing exactly one point is done as follows: First, partition all terminal pairs by the point they contain. This yields a series of sub-instances each centered on a common point.

Even though these sub-instances will generally overlap and their interactions have to be considered when trying to compute an optimal solution, they can be considered independent at the cost of a worse approximation factor: Each bounding box contains a grid point. If the width of a bounding box would be 2γ or larger, then along the x-axis, on the height of the original grid point, the bounding box had to contain another point, which is not allowed. For this reason, each bounding box has width of less than 2γ, the same is true for the height.

This leads to each sub-instance being fully contained within the four grid cells adjacent to the sub-instance inducing grid point. In turn, this limits the number of overlapping

sub-instances to a constant, allowing independent consideration of each individual sub-instance, which is illustrated in Figure 3.2.

There is a factor 4 approximation for each individual sub-instance. To do this, divide the terminal pairs into two kinds: those where the leftmost point is below the right one, and those where the left point is equal height to it, or above. Then, 2-approximate RSA for each of these sets, connecting each terminal point to the center. That this also yields a 2-approximation for GMMN has been discussed previously. Combining the sets yields a factor 4 approximation.

Merging the overlapping instances back together yields a factor 16 approximation.

And finally, merging this solution with the solution obtained for the terminal pairs containing two or more points, results in the final approximation factor.

Limited total area and large rectangles

Lemma 23. *For constant real numbers c, h, w, let P be a GMMN-Instance fully contained in a rectangle of size $w \times h$, where all bounding boxes of terminal pairs have height and width of at least c. The solution for P can then be factor 152 approximated in time $\mathcal{O}^*(2^{hwc^{-2}})$.*

By choosing a grid with side lengths w, h and line spacing of $c/2$, Theorem 22 can be used to approximate the solution.

Large rectangles

Theorem 24. *For constant real numbers c, C, with $c < C$, let P be a GMMN-Instance where all bounding boxes of terminal pairs have a height and width of at least c and at most C. The solution for P can be approximated up to a factor of* 166 *in time $\mathcal{O}^*(2^{C^2c^{-2}})$.*

Following the idea from the QPTAS for rectangle stabbing by Eisenbrand et al. [EGSV21], we subdivide the instance into a linear number of tiles, which can then be independently solved using Lemma 23.

To accomplish this, we will introduce evenly spaced vertical lines, that separate the terminal pairs into two variants: Those whose bounding box intersects such a line, and those whose bounding box doesn't. After solving the subinstance of all bounding boxes that were intersected, vertical strips of independent subinstances remain. This can then be repeated in the horizontal direction to obtain independent tiles.

However, it is not clear that the subinstance of the intersected bounding boxes can simply be solved separately, since these terminal pairs are not independent of the rest. To show that this is possible, we need the following lemma shown by Das et al. [DFK+18]:

Lemma 25. *x-separated 2D-GMMN admits, for any $\varepsilon > 0$, a $(6 + \varepsilon)$-approximation.*

Being x-separated is a property of an instance which requires that there is a vertical line that intersects all bounding boxes of terminal pairs. This is the case for each group of terminal pairs that were intersected by a common line, but solving these separately

without additional cost requires them to be independent. Let the separation between two vertical lines be $d > 2C$. This makes the separation distance at least twice as wide as the widest bounding box of a terminal pair. Since this is the case, the subinstances intersecting with different vertical lines must be independent from each other, and the entire intersecting subinstance can be approximated by approximating the subinstances around individual lines, employing Lemma 25.

Let $I_\mathrm{v}, I_\mathrm{h}$ be the subinstances that intersect the separating lines in the vertical and horizontal step, respectively. Approximate the solution for I_v, remove the pairs from P, approximate the solution for I_h, remove the pairs from P, and approximate the individual connected components of P with the approaches above. In total, the solution will cost at most $(6+\varepsilon)\cdot\mathsf{OPT}(I_\mathrm{v})+(6+\varepsilon)\cdot\mathsf{OPT}(I_\mathrm{v})+152\mathsf{OPT} <= 166\mathsf{OPT}$ for sufficiently small ε.

Since the maximum width and height of a connected instance is Cn each, this method generates at most $(\frac{Cn}{d})^2$ tiles which can be approximated in $\mathcal{O}^*(2^{d^2c^{-2}})$ each, which is $\mathcal{O}^*(2^{(C/c)^2})$ for $d \in \mathcal{O}(C)$.

4 Monotone Stabbing

Monotone Stabbing is a problem that seems to have not been defined before. It acts as a natural combination of the problems GMMN and Rectangle Stabbing: Monotone Stabbing (MSTAB) is the problem is finding a Generalized Manhattan Network for a set T of terminals and a set $S \subseteq T^2$ of pairs that are to be connected, which minimizes the total *horizontal* distance. This problem is like Stabbing on the bounding boxes of input pairs, except that rectangles do not have to be stabbed by one consecutive line segment: one can change the horizontal line segment in the middle of a rectangle, but only monotonically in a direction predefined for each rectangle.

We further simplify this problem by requiring that for all individual terminal pairs, the point with the greater x-coordinate is also the point with the greater y-coordinate, or in other words, that all connections go to the upper right. This is not a drastic change, since the original problem can be 2-approximated by separating the instance by direction of terminal pairs, solving both independently, and then combining the solutions.

Comparison with GMMN

While this problem is still nontrivial, and not a direct subcase of GMMN like MMN and RSA are, it is at most as hard as GMMN, and seems easier since there is only one dimension to optimize. This can be formally expressed as follows:

Theorem 26. *If there is a polynomial-time algorithm that approximates GMMN up to a factor k, there also is a polynomial-time algorithm that approximates MSTAB up to a factor k.*

Proof. This can be shown by solving arbitrary instances for MSTAB using an algorithm for GMMN. First, observe that scaling the y-axis does not change the horizontal distances of any Manhattan Network. By sufficiently scaling the given MSTAB instance, such that any vertical segments become negligible, it can directly be solved by GMMN:

Let P be a GMMN instance with terminals T and terminal point pairs C. The value of an optimal solution $\mathsf{OPT}(P)$ is upper bounded by $s := \sum_{\{p,q\} \in C} L_1(p, q)$, since there is a trivial solution with this exact length (connect all pairs $\{p, q\}$ by the path $p, (q.x, p.y), q$). The total length of all vertical line segments in an optimal solution is therefore also bounded by s.

Let G be the Hanan-Grid for the terminals. For any Manhattan Network M, there is another Manhattan Network that has the same length and L_1-connects the same set of terminals, but only uses line segments that are in G.

Let g be the length of the smallest line segment in the grid.

Given an instance P, find s and g for it, and scale all y-coordinates of the terminals by a factor of $g/(s+1)$ to produce the scaled instance P'. Since the sum of all vertical line segment lengths on any optimal solution was at most s, after the scaling it is now less than g.

The Manhattan Network M found by solving GMMN on P' is a valid solution for MSTAB on P': Let M be a solution on the grid w.l.o.g. Assume to contradict that M is not optimal for MSTAB. Then, there exists a better solution, that is, a Manhattan Network M' with a smaller total horizontal line length. But $|M'| \geq |M|$ still holds, since M is optimal for GMMN.

Consider the set $M \Delta M'$ of line segments that are in M, but not in M', or in M', but not in M. In this set, the length of vertical line segments has to be larger than the length of the horizontal segments.

But, by the choice of the scaling factor, the entire set of vertical line segments in M is smaller than g. Since the solutions are mapped to the grid, any horizontal segment has length at least g. And $M \Delta M'$ has to contain at least one horizontal line segment by the definition of M'.

Therefore, this is a contradiction, and the horizontal segments in M are in fact a valid solution for MSTAB on P'. Now, scaling back up also yields a valid solution for MSTAB on P. □

Since there is a known $\mathcal{O}(\log n)$ approximation of GMMN, the procedure used in Theorem 26 can be used to derive a $\mathcal{O}(\log n)$ approximation for MSTAB.

Corollary 27. *There is a factor $\mathcal{O}(\log n)$ approximation algorithm for MSTAB.*

Comparison with Stabbing

MSTAB seems simpler than GMMN, and one might assume that a Stabbing solution is only worse by a constant factor than the MSTAB solution on the same instance, but this is not the case in general:

Finding feasible solutions for MSTAB is simple, since Stabbing is known to have a factor 8 approximation using a dynamic program, and even a $(1+\varepsilon)$ approximation for all positive ε, and every solution for Stabbing also fits the criteria for an MSTAB solution: Even though jumps are now allowed, not using them is also valid. However, these solutions are not guaranteed to be optimal, since allowing jumps adds new valid solutions, which may be cheaper. It is in fact the case that there is a family of inputs, for which the Stabbing solution is at least a factor of $\log n$ worse than the optimal MSTAB solution. Since there is already an $\mathcal{O}(\log n)$-approximation algorithm for GMMN, and GMMN can solve MSTAB, using the normal Stabbing is not helpful in approaching MSTAB.

The instance family is illustrated in Figure 4.1 and can be inductively defined as follows:

I_0 is simply the unit rectangle. For all integral $n > 0$, given I_{n-1}, define I_n by placing two copies of I_{n-1} next to each other, so that one copy is strictly above and just to the right of the other. Then, also add the bounding rectangle encompassing both copies.

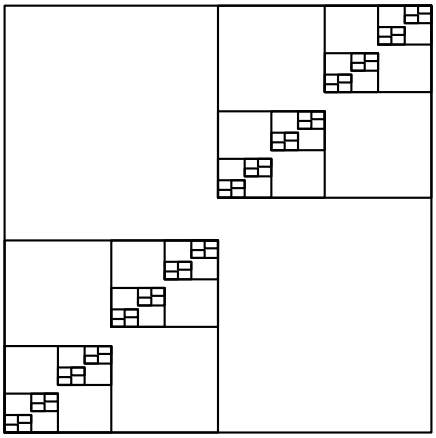


Fig. 4.1: A recursive family of instances for both Stabbing and MSTAB

There is a clear lower bound for the cost of stabbing an instance of this family: Consider instance I_d, for some $d > 0$. Every single one of the smallest rectangles, each with a width of 1, has to be stabbed. Since they never overlap each other, no line can ever be shared, so each of them needs to be stabbed individually, for a total cost of $2^d \cdot 2^0$. The next larger size of rectangles can reuse some of the line segments for the smallest rectangles, but since they do not share any y-coordinates, and each rectangle has to be stabbed by a single line, it can at most reuse the entire width of a small rectangle. Except for that, all rectangles of this size still need to be stabbed. This makes up a total cost of $2^{d-1} \cdot (2^1 - 2^0)$. Summing all of these segments, the overall best possible cost is

$$2^d + \sum_{i=1}^{d} 2^{d-i} \cdot (2^d - 2^{d-1}) = 2^d + \sum_{i=1}^{d} 2^{2d-i} - 2^{2d-i-1} = 2^{d-1}(2^d - 1) + 2^d$$

There is a monotone stabbing solution with size 2^d: As with Stabbing, each of the smallest rectangles needs to be covered individually, for the same cost of $2^d \cdot 1$. However, because the line is allowed to shift, this already stabs all larger rectangles as well.

Therefore, the approximation ratio of this approach is lower bounded by $(2^{d-1}(2^d - 1) + 2^d)/2^d = 1 + 2^{d-2}$, which grows arbitrarily.

Alternative Variant

The problem can be simplified further to a variant of Stabbing where changing line segments is always allowed, dropping the monotonicity requirements. However, this problem is easily exactly solvable in polynomial time: Split all input rectangles at the x-coordinates of the terminals, then solve the stabbing for each column of the input using the geometric hitting set algorithm, which runs in polynomial time, so the overall runtime is also polynomial.

This approach yields a valid solution, since in each strip each rectangle is hit by a line segment, and a sequence of such segments combines into a valid stabbing sequence. Any

optimal solution can be shifted to lie on the Hanan grid, and all solutions on the Hanan grid are considered by the algorithm, so the solution it returns is optimal.

Linear Programs

LP-Formulation 1

MSTAB itself is not trivial, since as with GMMN, local optimizations may pessimize the overall solution. However, the problem is relatively straightforward to express using mixed integer programming (*MIP*), which allows to take advantage of common optimization techniques.

The first formulation will be directly based on the problem definition, while the second formulation will take advantage of a known MIP statement of MMN.

In order to reduce the solution space to something which can be captured by a few variables, an additional fact is needed:

Lemma 28. *If M is an optimal solution of MSTAB on an instance I, there is another solution M' with the same cost as M, where M' only consists of line segments containing the lower left points of input rectangles.*

Proof. Assume M is an optimal solution of MSTAB on an instance I, in which no line segment can be moved downwards without invalidating the solution. Pick a line segment l from M which does not contain the lower left point of any rectangle of I. The only way to invalidate the solution by moving down the line segment a sufficiently small distance $\varepsilon > 0$ is if it either leaves a rectangle or passes another line segment.

If it would pass another line segment, then the segments can be merged into a longer segment, which can be repeated until a feasible solution M' is obtained, or the other case is encountered. The other case, however, can never be encountered, since it leads to a contradiction:

If a line segment leaves a rectangle for any shift ε, then the line segment was on the lower boundary of a rectangle. If this shift invalidates the solution, the line segment was necessary to stab the rectangle it is being shifted out of. However, to take part in stabbing a rectangle, the line segment l must either intersect the left boundary of the relevant rectangle directly, or be reachable via a shift from a line segment that does. It can never be reached from the left boundary of the rectangle by a shift, since l is at the bottom boundary.

Therefore, l also has to contain the left boundary, which is a contradiction to the selection of l. □

Let L be a constant at least as large as the width of the input instance, and n the number of rectangles in the input. Let $[n]$ name the set $\{1, 2, \ldots, n\}$. Define for each rectangle $i \in [n]$ the constants $X_i^-, X_i^+, Y_i^-, Y_i^+$ to denote the minimum/maximum x/y coordinate respectively. Let, without loss of generality, the rectangles being ordered by ascending Y_i^-.

For each rectangle $i \in [n]$, introduce the variables x_i^- and x_i^+ to denote the start and the end of a line segment running through the lower left vertex of i. For every pair

of rectangles (i, j), introduce the binary variables $c_{i,j}$ and $t_{i,j}$, which imply direct and indirect reachability between different line segments:

$$\text{variables} \qquad x_i^-, x_i^+ \in \mathbb{R} \qquad \forall i \in [n] \tag{4.1}$$

$$c_{i,j}, t_{i,j} \in \{0, 1\} \qquad \forall i, j \in [n] \tag{4.2}$$

$$\text{minimize} \quad \sum_{i=1}^{n} (x_i^+ - x_i^-) \tag{4.3}$$

$$\text{subject to} \qquad x_i^- \leq X_i^- \qquad \forall i \in [n] \tag{4.4}$$

$$X_i^- \leq x_i^+ \qquad \forall i \in [n] \tag{4.5}$$

$$-L + Lc_{i,j} + x_j^- \leq x_i^+ \qquad \forall i, j \in [n], i \leq j \tag{4.6}$$

$$\max(c_{i,j}, \max_{i<k<j}(t_{i,k} + c_{k,j} - 1)) = t_{i,j} \qquad \forall i, j \in [n], i \leq j \tag{4.7}$$

$$\max\{-L + Lt_{i,j} + x_j^+ \mid j \in [n], Y_k^- \leq Y_k^- \leq Y_j^+\} \geq X_i^+ \qquad \forall i \in [n] \tag{4.8}$$

This MIP considers only solutions that have the form of M' of Lemma 28. Based on that lemma, this will always include optimal solutions, so solution quality is not sacrificed by this limitation. Each potential line segment belonging to rectangle i is defined as running from $x_{i/l}$ to $x_{i/r}$ at height Y_i^-. A given rectangle may not have any associated line segment, in which case $x_{i/l} = x_{i/r} = x_i^-$, which contributes a cost of exactly zero to the target function, which is simply all the lengths of the individual line segments summed up.

In order to ensure that all these line segments form a shifted monotone stabbing of the input instance, there are a number of constraints. The main idea here is that t_{ij} can only be set to 1, if there is some way to use monotone jumps to reach line segment j starting from line segment i: If this is direct, that is, if line segment j starts to the left of the end of line segment i, then c_{ij} can be 1 (inequality 4.6). Only when there is either $c_{ij} = 1$, or there is a k such that $t_{ik} = 1 \wedge c_{kj} = 1$, then t_{ij} may be 1 (inequality 4.7).

Now, the condition of a correct shifted stabbing is easy to phrase: If, for each rectangle i, there is some sequence of monotone jumps from the lower left corner of i to a line segment, which is within i vertically, but extends to the right of i horizontally, then i is stabbed. This almost directly translates to constraint 4.8: The rightmost, monotonely reachable point just has to lie strictly to the right of i.

Note that, since the max operations are not optimized over, even after relaxing the integrality constraint of c_{ij} and t_{ij}, the solution space remains concave. The maximum is implementable in MIP, but not in LP, so there are further binary variables hidden in these constraints: Since the maximum is not linearly expressible, it gets split into the following linear inequalities. In general, the inequality $\max(x_1, x_2) \geq y$ can be

implemented using a binary variable b and an unconstrained variable m:

$$\begin{aligned} x_1 &\leq m \\ x_2 &\leq m \\ x_1 + Lb &\geq m \\ x_2 + L(1-b) &\geq m \\ m &\geq y \end{aligned}$$

This forces m to be the maximum of x_1 and x_2: It needs to be at least as large as they are, and depending on the value of b also equal to one of them. For maximum operations with more operands they can be rewritten as a nested form $\max(x_1, x_2, \ldots, x_k) = \max(x_1, \max(x_2, \max(..., x_k)))$ and then transformed individually.

LP-Formulation 2

Here, the infinite solution space is reduced by limiting the inspected line segments to the Hanan-Grid. The correctness of solutions is enforced by directing the grid and requiring a unit flow through each rectangle. The optimal solution is then found by minimizing over the total length of segments with nonzero flow.

Let G with vertices $V(G)$ and edges $E(G)$ be the graph induced by the Hanan grid, after directing each edge to the top right. Define for each rectangle i a subgraph $G_i \subseteq G$ only containing edges and vertices contained within the area of i, and designate the lower-left corner vertex with $v_{i/l}$, and the upper right corner vertex with $v_{i/r}$. For each vertex v of G, $\text{in}(v)$ is the set of incoming edges, and $\text{out}(v)$ is the set of outgoing edges.

For each terminal point pair, a separate flow problem will be solved. To this end, we define variables $x_{i,e}$ for each terminal pair i and for each edge e of G_i, which correspond to the flow across the edge. To make sure that there will only be one path taken by the flow, the variables have integer constraints. GMMN can be modeled using the following MIP:

$$\text{variables} \qquad x_{i,e} \in \{0,1\} \qquad \forall i \in [n], e \in E(G_i) \tag{4.9}$$

$$\text{minimize} \qquad \sum_{e \in E(G)} \max_{i \in [n]} \{x_{i,e}\} \tag{4.10}$$

$$\text{subject to} \qquad \sum_{e \in \text{in}(v)} x_{i,e} = \sum_{e \in \text{out}(v)} x_{i,e} \qquad \forall i \in [n], v \in V(G_i) \setminus \{v_{i/l}, v_{i/r}\} \tag{4.11}$$

$$\sum_{e \in \text{out}(v_{i/l})} x_{i,e} = 1 \qquad \forall i \in [n] \tag{4.12}$$

$$\sum_{e \in \text{in}(v_{i/r})} x_{i,e} = 1 \qquad \forall i \in [n] \tag{4.13}$$

Fig. 4.2: MSTAB instance which shows a big integrality gap for large L

By ignoring the vertical contribution to the objective function, this turns into a MIP for MSTAB. Let $E_{\text{hor}}(G)$ be the set of all horizontal edges of G:

$$\text{minimize} \quad \sum_{e \in E_{\text{hor}}(G)} \max_{i \in [n]} \{x_{ie}\} \tag{4.14}$$

Variables and constraints remain as above. Since the maximum function appears in the objective in this version, there is no need to introduce additional binary variables.

LP-Relaxation

Phrasing a problem as a mixed integer problem can be interesting, since it can sometimes directly produce approximation algorithms: By simply dropping all integrality constraints, the original MIP is *relaxed* into a proper linear program. Since linear programs can be solved efficiently, if one can find some way to efficiently turn the relaxed solution back into an integral one, this is a strategy to solve the problem. In general, this is only possible by sacrificing some solution quality, which is lower bounded by the *integrality gap*, which is defined as the worst possible ratio between an integral and a relaxed solution for the same input. In this case, for sufficiently big L, the integrality gap can be shown to be arbitrarily large:

For each $w \in \mathbb{N}$ define an MSTAB instance I_n as a rectangle from $(0, 0)$ to $(w, 5)$ and a rectangle from $(w - 1, 1)$ to $(w + 1, 3)$ as depicted in Figure 4.2.

The optimal solution for this instance has cost $n + 1$ and consists of the single line $[0, w + 1] \times \{1\}$. The LP-relaxation produces solutions with cost 2, for any w, as we will show now: Substituting the problem instance into the first linear program yields the following inequalities:

$$
\begin{aligned}
x_1^- &\leq 0 \leq x_1^+ \\
x_2^- &\leq w-1 \leq x_2^+ \\
-L + Lc_{1,2} + x_2^- &\leq x_1^+ \\
t_{0,1} &= c_{0,1} \\
x_2^+ &\geq w+1 \\
\max(-L + Lt_{1,2} + x_2^+, x_1^+) &\geq w
\end{aligned}
$$

Now, we apply the transformations to express the max function to the inequalities above:

$$
\begin{aligned}
x_1^- &\leq 0 \leq x_1^+ \\
x_2^- &\leq w-1 \leq x_2^+ \\
-L + Lc_{1,2} + x_2^- &\leq x_1^+ \\
t_{1,2} &= c_{1,2} \\
x_2^+ &\geq w+1 \\
x_1^+ &\leq m \\
-L + Lt_{1,2} + x_2^+ &\leq m \\
x_1^+ + Lb &\geq m \\
-L + Lt_{1,2} + x_2^+ + L(1-b) &\geq m \\
m &\geq w
\end{aligned}
$$

For this, the following variables can be chosen to still fulfill all inequalities, assuming $L \geq 2n-2$:

$$
x_1^- = x_1^+ = 0, \quad x_2^- = w-1, \quad x_2^+ = w+1, \quad c_{1,2} = t_{1,2} = \frac{w-1}{L}, \quad m = w, \quad b = \frac{w}{L}
$$

This leads to an objective function value of $x_1^+ - x_1^- + x_2^+ - x_2^- = 2$. Since this solves the system, and the optimal relaxed solution has a cost of at most the cost of this solution, while the optimal integer solution has an objective value of $w+1$, the approximation ratio obtained by this is lower bounded by $\frac{w+1}{2}$, which grows arbitrarily. Limiting L to a lower value is not an option in general, since the lowest value that does not admit invalid solutions even in the integer case is the width of the instance, which can be increased by copying several subinstances next to each other, which does not change their approximation factor.

5 Conclusion

Summary

There are constant-factor approximation algorithms for GMMN instances where all terminal pair bounding boxes share a common point, or where every bounding box contains a point of a constant-sized grid.

The shown special cases combine to cover a wide range of instances of GMMN: Given a connected instance with n terminal point pairs and a combined bounding box of size at least 1×1, the terminal pairs with a bounding box size of at most $\frac{1}{\sqrt{n}}$ can be naïvely connected for a constant increase in approximation factor.

The subinstance of pairs that have sufficiently large bounding boxes in both dimensions can be approximated by using the grid approach. Of the remaining pairs, the ones with a very large or very small ratio between width and height can be approximated by considering the shorter dimension to be irrelevant and treating it as a MSTAB instance, if MSTAB admits a constant-factor approximation.

New Open Problems

While there is a proof for GMMN, it is still unclear whether MSTAB is even $\mathcal{NP}$-hard, and if it is, if it admits polynomial time approximation schemes or constant-factor approximations. MSTAB also might be as hard as GMMN is in the sense that a constant-factor approximation algorithm of GMMN can be given in terms of an algorithm that solves MSTAB.

If MSTAB can be approximated, the case remaining to approximate GMMN is an instance consisting of rectangles which are neither very large nor very small, with a bounded aspect ratio. This also certainly is not trivial, but it might be more approachable than GMMN in general.

If for a GMMN instance, one can construct a point set such that each rectangle contains at least one point, and at most a constant number of subinstances induced by containing different points overlap, then this would also imply a constant-factor approximation of GMMN. While not possible in general, this may be a useful approach for some subproblem.

Bibliography

[BWWS06] Marc Benkert, Alexander Wolff, Florian Widmann, and Takeshi Shirabe: The minimum manhattan network problem: Approximations and exact solutions. *Comput. Geom.*, 35(3):188–208, 2006, 10.1016/j.comgeo.2005.09.004. https://doi.org/10.1016/j.comgeo.2005.09.004.

[CGS11] Francis Y. L. Chin, Zeyu Guo, and He Sun: Minimum manhattan network is np-complete. *Discret. Comput. Geom.*, 45(4):701–722, 2011, 10.1007/s00454-011-9342-z. https://doi.org/10.1007/s00454-011-9342-z.

[CNV08] Victor Chepoi, Karim Nouioua, and Yann Vaxès: A rounding algorithm for approximating minimum manhattan networks. *Theor. Comput. Sci.*, 390(1):56–69, 2008, 10.1016/j.tcs.2007.10.013. https://doi.org/10.1016/j.tcs.2007.10.013.

[CvDF+18] Timothy M. Chan, Thomas C. van Dijk, Krzysztof Fleszar, Joachim Spoerhase, and Alexander Wolff: Stabbing rectangles by line segments - how decomposition reduces the shallow-cell complexity. In Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao (editors): *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, volume 123 of *LIPIcs*, pages 61:1–61:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 10.4230/LIPIcs.ISAAC.2018.61. https://doi.org/10.4230/LIPIcs.ISAAC.2018.61.

[DFK+18] Aparna Das, Krzysztof Fleszar, Stephen G. Kobourov, Joachim Spoerhase, Sankar Veeramoni, and Alexander Wolff: Approximating the generalized minimum manhattan network problem. *Algorithmica*, 80(4):1170–1190, 2018, 10.1007/s00453-017-0298-0. https://doi.org/10.1007/s00453-017-0298-0.

[DGK+15] Aparna Das, Emden R. Gansner, Michael Kaufmann, Stephen G. Kobourov, Joachim Spoerhase, and Alexander Wolff: Approximating minimum manhattan networks in higher dimensions. *Algorithmica*, 71(1):36–52, 2015, 10.1007/s00453-013-9778-z. https://doi.org/10.1007/s00453-013-9778-z.

[EGSV21] Friedrich Eisenbrand, Martina Gallato, Ola Svensson, and Moritz Venzin: A QPTAS for stabbing rectangles. *CoRR*, abs/2107.06571, 2021. https://arxiv.org/abs/2107.06571.

[FS08] Bernhard Fuchs and Anna Schulze: A simple 3-approximation of minimum manhattan networks. In *Seventh Cologne Twente Workshop on Graphs and Combinatorial Optimization, gargano, Italy, 13-15 May, 2008*, pages 26–29. University of Milan, 2008.

[FS14] Stefan Funke and Martin P Seybold: The generalized minimum manhattan network problem (gmmn)–scale-diversity aware approximation and a primal-dual algorithm. In *Proceedings of Canadian Conference on Computational Geometry (CCCG)*, volume 26, 2014.

[GLN01] Joachim Gudmundsson, Christos Levcopoulos, and Giri Narasimhan: Approximating a minimum manhattan network. *Nord. J. Comput.*, 8(2):219–232, 2001. `http://www.cs.helsinki.fi/njc/References/gudmundssonln:219.html`.

[GNS16] Joachim Gudmundsson, Giri Narasimhan, and Michiel H. M. Smid: Geometric spanners. In *Encyclopedia of Algorithms*, pages 846–852. 2016, 10.1007/978-1-4939-2864-4_167. `https://doi.org/10.1007/978-1-4939-2864-4_167`.

[Han66] Maurice Hanan: On steiner's problem with rectilinear distance. *SIAM Journal on Applied mathematics*, 14(2):255–265, 1966.

[KSW21] Arindam Khan, Aditya Subramanian, and Andreas Wiese: A ptas for the horizontal rectangle stabbing problem, 2021.

[Maß15] Jens Maßberg: The depth-restricted rectilinear steiner arborescence problem is np-complete. *CoRR*, abs/1508.06792, 2015. `http://arxiv.org/abs/1508.06792`.

[MOY21] Yuya Masumura, Taihei Oki, and Yutaro Yamaguchi: Dynamic programming approach to the generalized minimum manhattan network problem. *Algorithmica*, 83(12):3681–3714, 2021, 10.1007/s00453-021-00868-x. `https://doi.org/10.1007/s00453-021-00868-x`.

[MSU09] Xavier Muñoz, Sebastian Seibert, and Walter Unger: The minimal manhattan network problem in three dimensions. In Sandip Das and Ryuhei Uehara (editors): *WALCOM: Algorithms and Computation, Third International Workshop, WALCOM 2009, Kolkata, India, February 18-20, 2009. Proceedings*, volume 5431 of *Lecture Notes in Computer Science*, pages 369–380. Springer, 2009, 10.1007/978-3-642-00202-1_32. `https://doi.org/10.1007/978-3-642-00202-1_32`.

[Ram03] Sarnath Ramnath: New approximations for the rectilinear steiner arborescence problem [VLSI layout]. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 22(7):859–869, 2003, 10.1109/TCAD.2003.814249. `https://doi.org/10.1109/TCAD.2003.814249`.

[Sch15] Michael Schnizler: The generalized minimum manhattan network problem. Master's thesis, 2015.

[SS05] Weiping Shi and Chen Su: The rectilinear steiner arborescence problem is np-complete. *SIAM J. Comput.*, 35(3):729–740, 2005, 10.1137/S0097539704371353. https://doi.org/10.1137/S0097539704371353.

[SU05] Sebastian Seibert and Walter Unger: A 1.5-approximation of the minimal manhattan network problem. In Xiaotie Deng and Ding-Zhu Du (editors): *Algorithms and Computation, 16th International Symposium, ISAAC 2005, Sanya, Hainan, China, December 19-21, 2005, Proceedings*, volume 3827 of *Lecture Notes in Computer Science*, pages 246–255. Springer, 2005, 10.1007/11602613_26. https://doi.org/10.1007/11602613_26.

[Zac00] Martin Zachariasen: On the approximation of the rectilinear steiner arborescence problem in the plane. *Unpublished manuscript, see http://citeseerx. ist. psu. edu/viewdoc/summary*, 2000.

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Würzburg, den 14.02.2022

. . . . . .
Tim Gerlach