Minimum Manhattan Network is NP-Complete

Francis Y. L. Chin
Department of Computer Science
The University of Hong Kong
Hong Kong, China
chin@cs.hku.hk

Zeyu Guo
School of Computer Science
Fudan University
Shanghai, 200433, China
gzy@fudan.edu.cn

He Sun
School of Computer Science
Shanghai Key Lab of IIP
Fudan University
Shanghai, 200433, China
sunhe@fudan.edu.cn

ABSTRACT

A *rectilinear path* between two points $p, q \in \mathbb{R}^2$ is a path connecting p and q with all its line segments horizontal or vertical segments. Furthermore, a *Manhattan path* between p and q is a rectilinear path with its length exactly $\text{dist}(p, q) := |p.x - q.x| + |p.y - q.y|$.

Given a set T of n points in $\mathbb{R}^2$, a network G is said to be a *Manhattan network* on T, if for all $p, q \in T$ there exists a Manhattan path between p and q with all its line segments in G. For the given point set T, the *Minimum Manhattan Network* (MMN) Problem is to find a Manhattan network G on T with the minimum network length.

In this paper, we shall prove that the decision version of MMN is strongly NP-complete, using the reduction from the well-known 3-SAT problem, which requires a number of gadgets. The gadgets have similar structures, but play different roles in simulating the 3-SAT formula. The reduction has been implemented with a computer program.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms

Theory

Keywords

Minimum Manhattan Network, 3-SAT, NP-complete

1. INTRODUCTION

1.1 Problem Description

A *rectilinear path* between two points $p, q \in \mathbb{R}^2$ is a path connecting p and q with only horizontal or vertical line segments as edges in the path. Furthermore, a *Manhattan path* between p and q is a rectilinear path with its length exactly equal to $\text{dist}(p, q) := |p.x - q.x| + |p.y - q.y|$, where $p.x$ ($q.x$) and $p.y$ ($q.y$) are the x and y coordinates of p (q) on the grid, *i.e.*, the Manhattan distance between p and q.

Given a set T of n points in $\mathbb{R}^2$, a network G is said to be a *Manhattan network* on T, if for all $p, q \in T$ there exists a Manhattan path between p and q with all its line segments in G. For a given network G, let the length of G, expressed by $L(G)$, be the total length of all line segments in G. For a given point set T, the *Minimum Manhattan Network* (MMN) Problem is to find a Manhattan network G on T with minimum $L(G)$.

From the problem description, it is easy to show that there is a close relationship between the MMN and planar t-spanners. For $t \geq 1$, a planar graph G is said to be a t-spanner of T if for all $p, q \in T$, there exists a path in G connecting p and q of length at most t times the distance between p and q. The MMN Problem for T is exactly the problem to compute the shortest 1-spanner of T under the L_1-norm [3, 4].

1.2 Historical Review

Due to numerous applications in city planning, network layout, distributed algorithms, and VLSI circuit design, the MMN problem was firstly introduced in 1999 by J. Gudmundsson et al. [4]. In that paper, they proposed an $O(n^3)$-time 4-approximation algorithm, and an $O(n \log n)$-time 8-approximation algorithm. Especially, they highlighted three open problems: (1) whether or not MMN is NP-hard; (2) whether or not PTAS exists for MMN; and (3) whether or not a 2-approximation algorithm exists for MMN.

After [4], much research was devoted to finding approximation algorithms for the MMN problem. Most combinatorial constructions [1, 2, 5, 9] rely on the decomposition of the input, by partitioning the input into several blocks (orthoconvex regions) that can be solved independently. R. Kato et al. [7] presented an $O(n^3)$-time 2-approximation algorithm. Although the correctness proof of their algorithm is incomplete [3], the paper showed that checking the existence of a Manhattan path for $O(n)$ specific pairs of points, instead of $O(n^2)$ pairs in $T \times T$, would be sufficient for determining whether the graph was a Manhattan network. Following this idea, M. Benkert et al. [1, 2] proposed an $O(n \log n)$-time 3-approximation algorithm. They also described a mixed-integer programming (MIP) formulation of the MMN problem. After that, V. Chepoi et al. [3] used the notion *Pareto Envelope* and nice strip-staircase decomposition to divide the plane into several regions, which can be studied individ-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SCG'09, June 8–10, 2009, Aarhus, Denmark.
Copyright 2009 ACM 978-1-60558-501-7/09/06 ...$5.00.

ually. Based on this idea, they proposed a 2-approximation rounding algorithm by solving the linear programming relaxation of the MIP. In K. Nouioua's Ph.D thesis [8], a primal-dual based 2-approximation algorithm with running time of $O(n \log n)$ was presented. Later, Z. Guo et al. [5] observed that the same approximation ratio can also be achieved with time complexity $O(n^2)$ using a combinatorial construction, in particular using the dynamic programming speed-up technique of quadrangle inequality. Furthermore, Z. Guo et al. [6] presented a simple 2-approximation algorithm for constructing a Manhattan network with $O(n \log n)$ running time. Compared with the previously best known combinatorial construction [5], one highlight of [6] is that, it was proved that a simple greedy strategy was able to construct a 2-approximation MMN. S. Seibert et al. [9] proposed a 1.5-approximation algorithm. However, their proof may be incorrect [3]. Despite the wealth of publications on the MMN problem, prior to our work it was open whether MMN is NP-hard.

1.3 Our Approach and Results

In this paper, we shall prove that the decision version of the MMN problem is strongly NP-complete, using the reduction from the well-known 3-SAT problem, which requires a number of gadgets. The gadgets have similar structures, but play different roles in simulating the 3-SAT formula.

The construction of the gadgets and the reduction rely on the following ideas: (1) A horizontal or vertical line always exists between two points having the same x-coordinate or y-coordinate (Figure 1a). (2) There are many ways of connecting two points with different x-coordinates and y-coordinates, but we are only interested in the two ways which use exactly one horizontal and one vertical line segment. These two ways of connecting two points will be used to represent the assignment of 1 and 0 to a boolean variable (Figure 1b). (3) There are several ways of connecting line segments to staircase points. These different ways of connecting staircase points form the bases of different gadgets with various functions (Figure 1c).

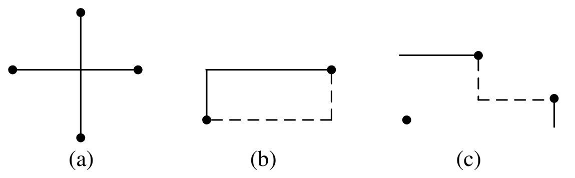

Figure 1: Intuition behind our construction

Our reduction generates a point set T from any boolean formula ψ. There are three types of edges in a Manhattan network on T, which are generated based on the three above-mentioned ideas, *i.e.* $G = (T, E_F \cup E_S \cup E_C)$. E_F consists of all line segments $[p, q]$ based on the first idea, with $p, q \in T, p.x = q.x$ or $p.y = q.y$. Basically E_F contains those line segments whose endpoints have the same x-coordinates (or y-coordinates) and form the basic structure of all gadgets; E_S are set of strip paths based on the second idea, and each path usually consists of one vertical and one horizontal line segment, connecting two points of different coordinates; E_C consists of line segments within the gadgets. Thus the total length of E_F and E_S should be fixed and only depend on ψ. What matters will be how strips are connected, *i.e.* variables are assigned, and this will affect the total length of E_C. E_C are line segments connecting one or two isolated staircase points in each gadget to other points in the same gadget based on the third idea. Let ℓ_α be the length sum of these line segments in each gadget α, which will depend on the assignments of those strips (variables) associated with that gadget and this is used to enforce relationship of these assignments in each gadget. The total length of E_C can be computed by $\sum_\alpha \ell_\alpha$ and minimum if each ℓ_α can achieve the minimum value. Unfortunately the values of ℓ are not independent, an assignment of a strip that makes ℓ_α minimum might not be minimum for ℓ_β where $\beta \neq \alpha$. For ease of evaluation, we shall introduce a "potential cost" for each strip depending on its assignment and define cost_α be the sum of ℓ_α and potential cost of all strips associated with gadget α. We shall show that the value of E_C will be bounded by a certain value if and only if the assignments of all strips (variables) give minimum value cost_α for each gadget α. Furthermore, we can show that cost_α achieves minimum value for each gadget α if and only if ψ is satisfiable. Thus there exists a Manhattan network of overall length bounded by a certain value if and only if ψ is satisfiable.

As a consequence of our reduction, it is easy to show that MMN is strongly NP-complete and there does not exist FPTAS algorithm for this problem unless $P = NP$. This also answers the first problem of J. Gudmundsson et al. [4], which was open for more than ten years.

Section 2 gives the basic notations and concepts, followed by our reduction in Section 3. Section 4 is devoted to the correctness proof for the reduction.

2. PRELIMINARIES

Denote $R(p, q)$ as a closed rectangle where $p, q \in \mathbb{R}^2$ are its two opposite corners. $B_V(p, q)$ is defined as the vertical closed band bounded by p, q, *i.e.*

$$\Big\{(x, y) \mid \min\{p.x, q.x\} \le x \le \max\{p.x, q.x\}, y \in \mathbb{R}\Big\},$$

whereas $B_H(p, q)$ denotes the horizontal closed band bounded by p, q, *i.e.*

$$\Big\{(x, y) \mid \min\{p.y, q.y\} \le y \le \max\{p.y, q.y\}, x \in \mathbb{R}\Big\}.$$

Formally, for $p, q \in T, p.x < q.x, p.y < q.y$, we call $R(p, q)$ a *vertical strip* if there does not exist any point of T in the region $B_V(p, q)$ except on the vertical lines $\{(x, y)|x = p.x, y \le p.y\}$ and $\{(x, y)|x = q.x, y \ge q.y\}$ (represented by the dashed lines in Figure 2). The *length of a vertical strip* $R(p, q)$ is defined as $|p.y - q.y|$, and the *width of a vertical strip* is defined by $|p.x - q.x|$. *Horizontal strip* is defined similarly as shown in Figure 2. For any strip $R(p, q)$, we call a Manhattan path connecting p and q a *strip path* of $R(p, q)$. Moreover, for each strip we choose one Manhattan path and these paths compose set E_S, called *strip path collection.* The strip path collection E_S is said to be *nice* if, for each strip $R(p, q)$, the strip path connecting p and q in E_S lies on $\partial R(p, q)$, *i.e.*, the boundary of $R(p, q)$.

In our construction, each strip can be employed to express the value of a boolean variable, which is indicated by which side of the strip the path (network) includes. Specifically, from the definition of vertical strip, we can assume that the strip path of $R(p, q)$ would not contain any vertical line segment whose x-coordinate does not equal to $p.x$ or $q.x$ [10],

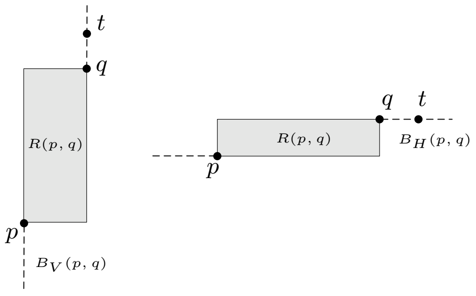

Figure 2: The rectangle is a vertical/horizontal strip $R(p,q)$. Any point in T within $B_V(p,q)$ or $B_H(p,q)$ can only be placed on the dashed lines, *e.g.*, point t.

thus has only one horizontal line segment (*switch segment*). For a nice strip path collection E_S, there are two ways of connecting p and q using a strip path in E_S, which can be used to express the value of variable v in boolean formula ψ. The nice property of a strip path is also used to transport the value of v from one end into the other. As shown in Figure 3, the solid line segments represent $v = 1$ and the dashed line segments for $v = 0$. In our reduction, we first assume that all strip paths are nice and, in Section 4, we shall show how to deal with other cases of strip paths.

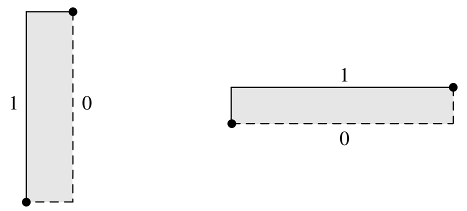

Figure 3: For each strip, two kinds of Manhattan paths are used to represent the assignment of variable v. The solid line segments represent $v = 1$, and the dashed line segments represent $v = 0$.

3. REDUCTION

3.1 Reduction Overview

We begin with a high-level overview of our reduction. The main task is to convert any given boolean formula ψ with n variables and m clauses into a point set T such that ψ is satisfiable if and only if the length of MMN G on T, expressed by $L(G)$, is bounded by a polynomial-time computable value.

As shown in Figure 4, each line segment, denoting one strip, is used to represent a literal (with vertical line for the vertical strip and horizontal line for horizontal strip), each circled node represents one of the gadgets, and each dotted-line square represents one of the clauses in ψ. Specifically, the two adjacent vertical lines numbered by $2i-1, 2i, 1 \leq i \leq n$, which might be segmented by the `SPLITTER` gadgets, are used to represent two literals x_i and $\neg x_i$ respectively. The upper end of each vertical line is initiated by a `DUMMY` gadget whereas the lower end of each vertical line is connected to a `NEGATOR` gadget or a `TURN` gadget so as to ensure that, in the network, the connection using 0 or 1 vertical line segment (corresponding to the assignment of a variable) can be swapped or maintained. These vertical lines (strips) and their associated gadgets are placed in such a way that they will not interfere with each other. The assignments of the variables have to be transported, through the `SPLITTER` gadget, to the `CLAUSE` gadget. The `CLAUSE` gadget is connected to three line segments, which correspond to three literals, and has a connection of lowest length in all assignments except one ($2^3 - 1$ assignments); this corresponds to the situation that the clauses will be satisfied for all but one of the assignments. The m dotted-line squares, each of which represents an individual clause and consists of five gadgets, are placed at the right-part of the network, again without interfering with each other and with other parts of the network. The `NEGATOR` gadget together with `CLAUSE` gadget is needed to ensure that the connection will achieve the lowest length if the clause is satisfiable. In other words, the Manhattan network will have the minimum length if and only if there exists an assignment to the variables such that the resultant assignment to every clause is satisfied. In order to construct the desired network, the strips involving in the construction have two different widths—standard width 20 and narrow width 10, and gadget `ADAPTOR` is for connecting a strip with narrow width to a strip of standard width.

3.2 Gadget Design

Our construction relies on six different gadgets, `NEGATOR, TURN, SPLITTER, CLAUSE, ADAPTOR`, and `DUMMY`, and the gadgets are placed in such a way that there exists a strip, either horizontal or vertical, between two gadgets (Figure 4). Generally speaking, each gadget (Figure 5) consists of three pairs of points, a pair of black points b_1, b_2 at the opposite corners of the gadget, a pair of white points w_1, w_2 and a pair of grey points g_1, g_2 which might be degenerated into one grey point. The positions of points inside each gadget are as follows:

$$b_1.x < w_1.x < g_1.x \leq g_2.x < w_2.x < b_2.x$$

$$b_1.y < w_2.y < g_2.y \leq g_1.y < w_1.y < b_2.y$$

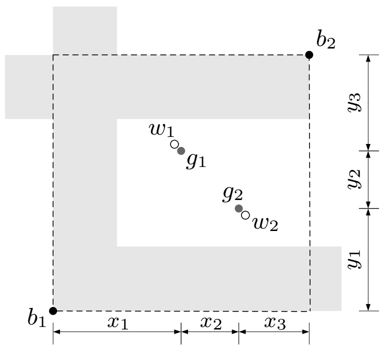

Figure 5: Configuration of gadgets. Generally speaking, each gadget consists of two black points, two white points and two grey ones. In the degenerated case, two grey points may coincide and the gadget only consists of five points. The grey regions represent three strips.

Two grey points are very close to the white points with distance ϵ under the L_1-norm, $\epsilon \leq \frac{1}{2n_G}$, where n_G is the

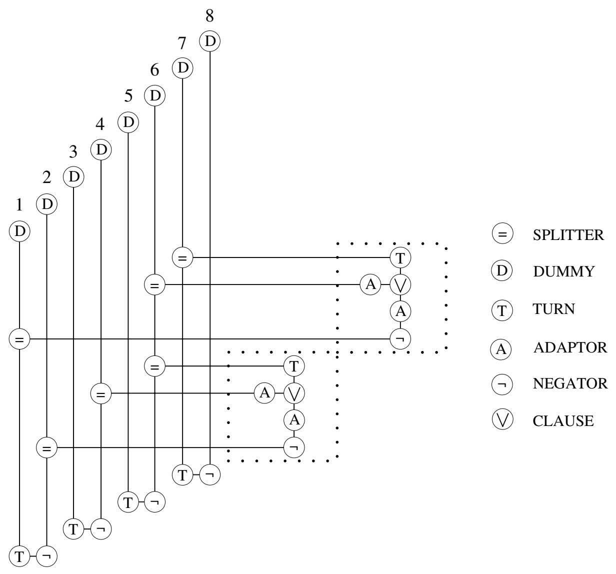

Figure 4: A schematic view of the construction. Each line segment represents a strip, and each circled node represents a gadget. On the left side of this figure, vertical strips are employed to represent the assignment of n variables in ψ, whereas on the right side of the figure, each square, consisting of five gadgets, corresponds to a clause. This graph corresponds to the input boolean formula $\psi = (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$.

number of gadgets and will be determined later. Notice that two grey points may coincide in the degenerated case. We shall assume that the length of the connection between w_1 and g_1, and similarly w_2 and g_2, is very small, much smaller than 1, and they can be ignored as long as the total length of MMN is within a certain value. For the following discussion of gadgets, the gadgets are placed in such a way that b_1 of a gadget and b_2 of another will form a strip. Note that b_1 will be always located at the lower left corner of the strip and b_2 the right upper corner. Depending on their relative positions, horizontal or vertical strips will be formed according to the schematic diagram of Figure 4. Moreover, each side of the strip will lie on the boundary of one gadget at one end and inside the other gadget at the other end (Figure 19). It is always advantageous to have the strip path be assigned inside the gadget, as this will shorten the connection length within the gadget. That is the intuitive reason (formal proof in Section 4) why the strip paths are always nice. Should the path cross over to the other side of the strip in the middle, it will be on the outside boundary of both gadgets, which will result in a larger connection lengths within both gadgets.

Let B be a big rectangle with sides parallel to the axes, and enclosing all the gadgets and strips. Assume the four boundary edges ∂B of B are on line $x = x_1, x = x_2, y = y_1, y = y_2$ respectively ($x_1 < x_2, y_1 < y_2$). Initially, let $T = T_0$ be the corner point set of B, *i.e.*,

$$T_0 := \{(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)\}.$$

Then for each gadget, we add eight points on ∂B

$$(x_1, b_1.y), (b_1.x, y_1), (x_2, b_2.y), (b_2.x, y_2),$$

$$(x_1, w_1.y), (w_1.x, y_2), (x_2, w_2.y), (w_2.x, y_1),$$

to T, as shown in Figure 6. When there is no strip along some side of a gadget, one point of ∂B should be added to T such that this point, together with b_1 or b_2, is on a vertical or horizontal line that goes along that side of the gadget without strip. For example, in Figure 6, there is no strip on the right side of the gadget, therefore point u is placed on the bottom edge of ∂B with its x-coordinate same as $b_2.x$. Similar construction can be applied for the other three sides. By adding these points, some line segments, actually in E_F and represented by the solid lines in Figure 6, are forced to be in any Manhattan network. Furthermore, the Manhattan network should contain paths between w_1 and g_1, g_1 and g_2, w_2 and g_2, and also between g_1, g_2 and b_1, and similarly between g_1, g_2 and b_2. For different gadgets, w_1, w_2, g_1 and g_2 are placed at different locations so that different relations for the assignments of the strips can be enforced for obtaining an MMN.

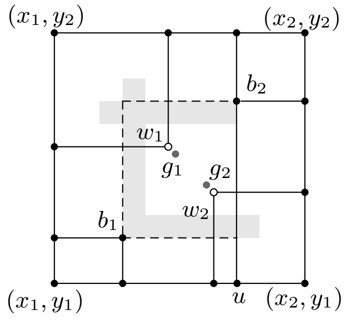

Figure 6: The extra points of ∂B with respect to one gadget. For each gadget, we add eight points in the resultant network. Based on the relative positions among different gadgets we add point u in the resulting network.

In the following, we shall describe the `NEGATOR` gadget in detail and outline the other gadgets. Since the distance

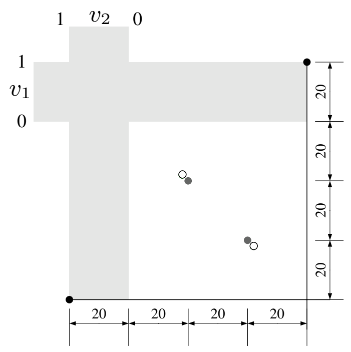

Figure 7: `NEGATOR`

Table 1: Cost Function of `NEGATOR` $\text{cost}_{\min} = 110$

v_1	v_2	ℓ	potential	cost
0	0	100	20	120
0	1	100	10	110
1	0	100	10	110
1	1	120	0	120

between g_1 and w_1, and similarly g_2 and w_2, is very small, we shall ignore this value in the following tables, and the sum of such distances for all the gadgets will be considered when calculating the total length of Manhattan network.

3.3 Gadget NEGATOR

The Manhattan network in each gadget is shorter in length when the strip path runs inside as opposed to outside (the boundary of) the gadget. This will make the analysis difficult if each gadget is studied in isolation, as each strip path should always run inside one gadget and outside on the other. For ease of analysis so that we can simply consider the length of each gadget with respect to the different assignments of the strips without worrying about their global effects on the other gadgets, each strip is associated with an extra *potential cost* which will be included in the cost calculation if the strip path is inside the gadget, *i.e.* on the left or top of the gadget ($v = 0$), or on the right or bottom of the gadget ($v = 1$). Informally, the potential cost of each strip will be added into the cost evaluation of gadget α if and only if the strip path is in the interior of α.

As shown in Figure 7, the `NEGATOR` gadget is involved with two strips, whose connection relates the assignments of the variables v_1 and v_2, and ℓ is the shortest length of line segments so that for each point pair in the gadget, except (w_1, g_1) and (w_2, g_2), there exists a Manhattan path. We would like the connection of the grey points in this gadget to have the lowest length if the assignments of v_1 and v_2 are different, *i.e.* this `NEGATOR` gadget will ensure that $v_1 = \neg v_2$ if the length of the network has to be within a certain value. In the following we consider the shortest connections of the grey points for the four different assignments of the two strips. The four possible connections are indicated by the dashed lines as shown from Figure 8 to Figure 11.

Besides the cases with $v_1 = \neg v_2$ which give the lowest length, the case $v_1 = v_2 = 0$ also gives the lowest length. However, careful analysis of the situation will show that for the case $v_1 = v_2 = 0$, the length of connection in the other

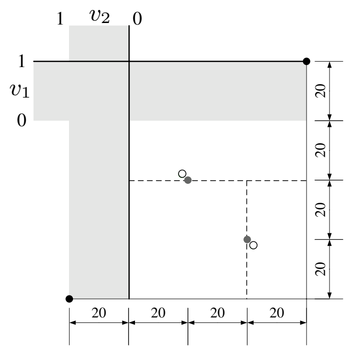

Figure 8: $v_1 = 1, v_2 = 0, \ell = 100$

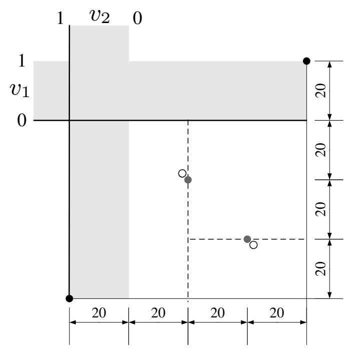

Figure 9: $v_1 = 0, v_2 = 1, \ell = 100$

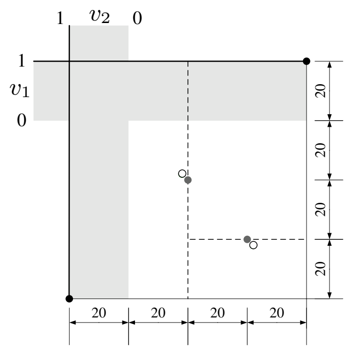

Figure 10: $v_1 = 1, v_2 = 1, \ell = 120$

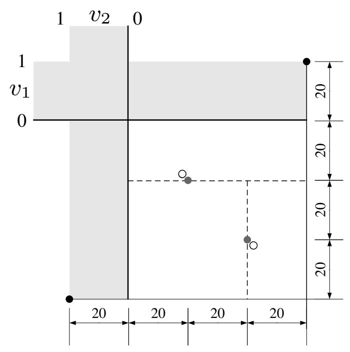

Figure 11: $v_1 = 0, v_2 = 0, \ell = 100$

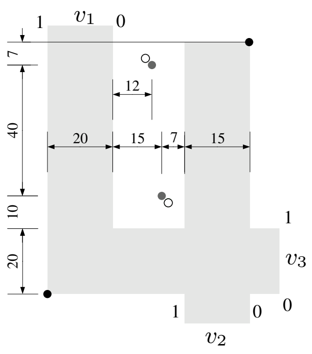

Figure 12: `SPLITTER`

Table 2: Cost Function of `SPLITTER` $\text{cost}_{\min} = 87$

v_1	v_2	v_3	ℓ	potential	cost
0	0	0	77	10	87
0	0	1	72	20	92
0	1	0	69	20	89
0	1	1	67	30	97
1	0	0	107	0	107
1	0	1	87	10	97
1	1	0	87	10	97
1	1	1	67	20	87

gadgets will not be lowest. An assignment of a strip always affects two gadgets and the same strip always locates at different sides of the two gadgets, at top side of one and bottom side of the other, similarly for the left and right sides. An assignment of 0 or 1 for a strip is good for one gadget, but will be bad for the other. Thus the assignment $v_1 = v_2 = 0$ would be bad for the other two gadgets and consequently the length of the resultant network for this case would be longer than the other two cases.

To represent this dependence, we introduce a potential cost associated with each strip. The potential cost of 10 for each strip can be added to exactly one of two gadgets adjacent to this strip when evaluating cost function. Since the number of strips is fixed for the input formula ψ, the sum of potential costs used will be subtracted latter and do not affect the final length of the network.

By including this potential cost, the cumulated costs for the four cases are as shown in Table 1, where the lowest costs occur when $v_1 = \neg v_2$. Similar analysis is applied for the other gadgets, with the total length listed in the tables.

3.4 Other Gadgets

The `SPLITTER` gadget can "generate" a horizontal branch from the original vertical strip and the generated horizontal strip has the same assignment with the original vertical strip. In order to generate such a branch, two vertical strips as well as one horizontal strip are located as shown in Figure 12. The choice of values involving in the gadget guarantees that $\text{cost}_{\min}$ is obtained if and only if all of the three strips get the same 0-1 value (Figure 12, Table 2).

Since originally the assignment for each variable is represented by vertical strips, the `TURN` gadget (Figure 13 and Figure 14) is used to change the direction of information flow so that the horizontal strip can represent the same assignment with the crossing vertical strip in the `TURN` gadget, and the $\text{cost}_{\min}$ in the `TURN` gadget is obtained if and only if the two crossing strips are expressing the same assignment (Figure 13 and 14, Table 3).

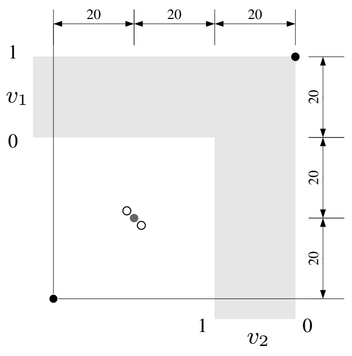

Figure 13: `TURN`**(a)**

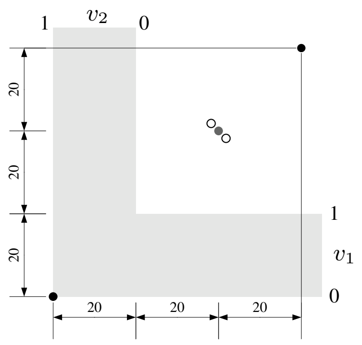

Figure 14: `TURN`**(b)**

Table 3: Cost Function of `TURN`**(a)** $\text{cost}_{\min} = 50$

v_1	v_2	ℓ	potential	cost
0	0	40	10	50
0	1	40	20	60
1	0	60	0	60
1	1	40	10	50

Three strips in the `CLAUSE` gadget represent the three literals associated with the clause, and through other gadgets, these strips are connected to the vertical strips on the left representing the individual literals. The design of the `CLAUSE` gadget is to make a gap in cost between the true assignments of a clause in ψ and the false one. In other words, $\text{cost}_{\min}$ has to be achieved for the `CLAUSE` gadget in seven of eight assignments of three variables (Figure 15, Table 4). Note that the cost function will not achieve minimum only for the case $v_1 = 0, v_2 = 1, v_3 = 0$. However, since one of three strips, actually representing v_2, is connected to the left vertical strip through the `NEGATOR` gadget, thus for the `CLAUSE` gadget, $\text{cost}_{\min}$ can be achieved if and only if the corresponding clause can be satisfied for the assignment of the input formula.

In order to achieve this goal, the width of two out of three strips in the `CLAUSE` gadget is set to 10 and smaller than the standard width 20. The `ADAPTOR` gadget (Figure 16 and

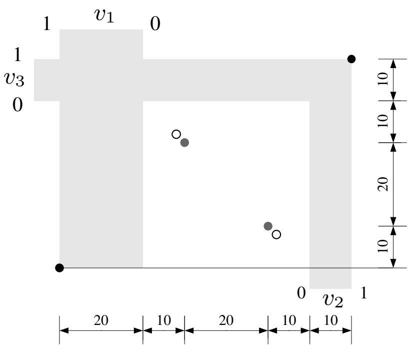

Figure 15: CLAUSE

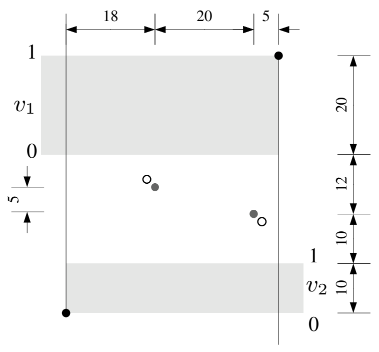

Figure 17: ADAPTOR(b)

Table 4: Cost Function of CLAUSE $\text{cost}_{\min} = 90$

v_1	v_2	v_3	ℓ	potential	cost
0	0	0	70	20	90
0	0	1	80	10	90
0	1	0	70	30	100
0	1	1	70	20	90
1	0	0	80	10	90
1	0	1	90	0	90
1	1	0	70	20	90
1	1	1	80	10	90

Table 5: Cost Function of ADAPTOR(a) $\text{cost}_{\min} = 65$

v_1	v_2	ℓ	potential	cost
0	0	55	10	65
0	1	68	0	68
1	0	47	20	67
1	1	55	10	65

Figure 17, Table 5) is used to connect two strips of different widths together, and $\text{cost}_{\min}$ is obtained if and only if these two strips represent the same assignment.

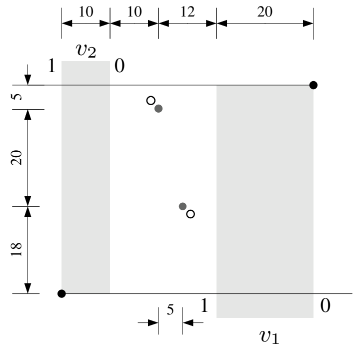

Figure 16: ADAPTOR(a)

To bound the vertical strip associated with each literal, we introduce the DUMMY gadget (Figure 18, Table 6).

3.5 Putting Them Together

According to the diagram of Figure 4, we combine different gadgets to form the desired network. For instance, in Figure 4 the SPLITTER gadget is connected to gadgets DUMMY, TURN, and NEGATOR. So we let the DUMMY gadget and SPLITTER gadget share a common vertical strip, the TURN gadget and SPLITTER gadget share a common vertical strip, whereas the horizontal strip of the SPLITTER gadget is connected to the NEGATOR gadget, as shown in Figure 19. Similar approach can be applied for connecting other gadgets and obtaining the point set T.

In addition, if gadget α is located on the left of gadget β in Figure 4, then all the points in α fall on the left of the points in gadget β. Similarly, if gadget α is located above gadget β in Figure 4, then all the points in α are located above the points in gadget β.

As described before, we can only consider the optimal network connection of each gadget without worrying about the global network length. Actually we shall prove in Section 4 that ψ is satisfiable if and only if $\text{cost}_{\min}$ can be obtained for each individual gadget, *i.e.* the total cost for the resulting network is bounded by a polynomial-time computable value. In summary, we have the following results.

LEMMA 1. $\text{cost}_{\min}$ *is obtained*

- *for any assignment of the strip in the* DUMMY *gadget.*
- *if the crossing strips represent different 0-1 values in the* NEGATOR *gadget.*
- *if all the crossing strips represent the same 0-1 value in the* ADAPTOR*,* TURN *and* SPLITTER *gadgets.*
- *for all seven of the eight assignments of the three variables on crossing strips in the* CLAUSE *gadget.*

4. ANALYSIS

As shown in Figure 4, it is easy to observe the following facts.

LEMMA 2. *For any boolean formula ψ with n variables and m clauses, there are m* CLAUSE *gadgets, $2m$* ADAPTOR *gadgets, $(m+n)$* TURN *gadgets, $(m+n)$* NEGATOR *gadgets, $3m$* SPLITTER *gadgets and $2n$* DUMMY *gadgets in the resulting instance of MMN.*

COROLLARY 1. *In the resulting instance reduced from the boolean formula ψ with n variables and m clauses, the number of gadgets*

$$n_G = 8m + 4n,$$

and the number of strips

$$n_S = 10m + 3n.$$

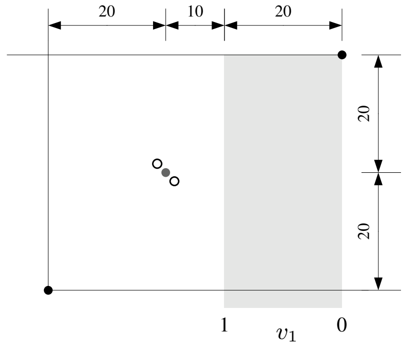

Figure 18: DUMMY

Table 6: Cost Function of DUMMY $\text{cost}_{\min} = 40$

v_1	ℓ	potential	cost
0	40	0	40
1	30	10	40

COROLLARY 2. *For the resulting instance reduced from the boolean formula ψ with n variables and m clauses,*

$$\sum \text{cost}_{\min} = 240n + 641m.$$

Define $\mathcal{Q}_k(p)$ be the k-th closed quadrant originated at point p, where $k = 1, 2, 3, 4$. For any point $p \in T \backslash T_0$ (note that T_0 contains the four corner points of the big rectangle B), let $h_k^x(p)$ be the point in $T \cap \mathcal{Q}_k(p) \backslash \{p\}$ whose x-coordinate is closest to $p.x$ (if more than one exist, choose the one whose y-coordinate is closest to $p.y$). Similarly, we can give the definition of $h_k^y(p)$. Figure 20(a) shows an example of $h_1^x(p)$ and $h_1^y(p)$.

LEMMA 3. *For a network G, if for any point $p \in T \backslash T_0$, there exist Manhattan paths connecting p and $h_k^x(p)$, as well as p and $h_k^y(p)$, $k = 1, 2, 3, 4$, then G is a Manhattan network on T.*

PROOF. It suffices to show that for any points $p, q \in T$ satisfying $T \cap R(p, q) \backslash \{p, q\} = \emptyset$, there exists the Manhattan path connecting p and q.

First, we discuss the case that $p \in T_0$ or $q \in T_0$. Without loss of generality, assume $p = (x_1, y_1)$ (Figure 6). Then $q \notin T_0$. We have $h_3^x(q) = p$ and p, q are connected by a Manhattan path.

Otherwise, assume $p.x \le q.x, p.y \le q.y$ and the other cases can be proven similarly. In such case, the path connecting p and $h_1^x(p)$ and the path connecting q and $h_3^y(q)$ intersect, and as shown in Figure 20(c), they form a Manhattan path connecting p and q (note that in the special case $h_1^x(p)$ and q, as well as $h_3^y(q)$ and p, can coincide). $\square$

The following theorem gives the necessary and sufficient conditions of a network G being a Manhattan network on T even though G might not be minimum.

THEOREM 1. *A network G is a Manhattan network on T if and only if G satisfies the following three properties: (1) $E_F \subseteq G$, (2) G contains a strip path collection, (3) G contains a Manhattan subnetwork on the vertex set of each gadget.*

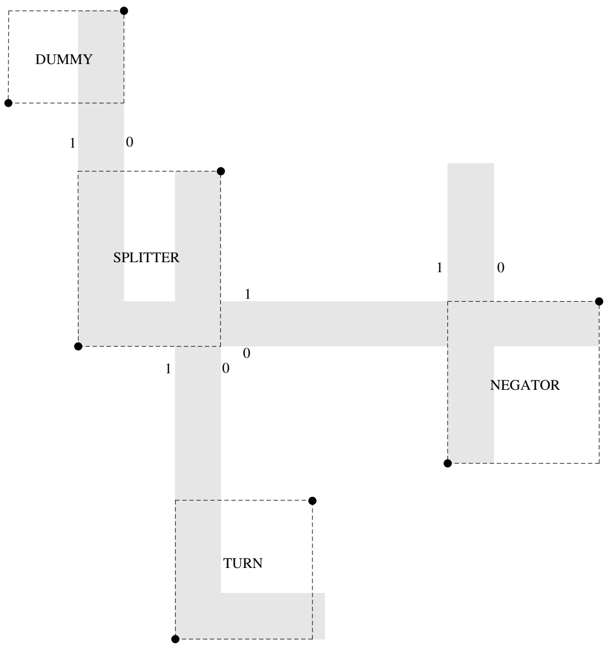

Figure 19: The method of connecting different gadgets. Two adjacent gadgets always share a common strip.

PROOF. It is easy to see that a Manhattan network must satisfy these three properties. So we only need to prove that any network G satisfying these conditions is a Manhattan network. By Lemma 3 it suffices to show that for any point $p \in T \backslash T_0$, there exists a Manhattan path connecting p and $h_k^x(p)$, as well as p and $h_k^y(p)$.

For symmetry, we only consider the case of $h_1^x(p)$ and $h_2^x(p)$. If there exists point $u \in T$ such that $u.x = p.x, u.y > p.y$, and there is no point of T between p and u, then $h_1^x(p) = h_2^x(p) = u$ and G contains a Manhattan path connecting p and $h_1^x(p)$, as well as p and $h_2^x(p)$ (Property 1). It is easy to verify that in such case p is one of points w_1, b_2 of each gadget, or the points on ∂B, except the corner and the top boundary ones.

For the point p lying on the top boundary of ∂B, it is easy to see that the top horizontal line $(y = y_2) \cap \partial B$ contains Manhattan paths connecting p and $h_1^x(p)$, as well as p and $h_2^x(p)$ (Property 1).

So the rest is to consider those cases when p is one of the points $\{b_1, g_1, g_2, w_2\}$ of any gadget α, where there does not exist point $u \in T$ satisfying $u.x = p.x, u.y > p.y$.

If $p = b_1$ of gadget α, denoted by b_1^α, then there exists gadget β such that $R(b_1^\alpha, b_2^\beta)$ is a vertical strip. As shown in Figure 20(b), $h_1^x(p) = b_2^\beta$ and $h_2^x(p) = w_2^\beta$. p and $h_1^x(p)$ is connected by a Manhattan path in the strip path collection (Property 2), whereas the Manhattan path between p and $h_2^x(p)$ is obtained by combining the horizontal line segment between p and the left boundary point $(x_1, p.y)$ with the vertical line segment with endpoints w_2^β and $(w_2^\beta.x, y_1)$ (Property 1).

For the other cases, when $p \in \{g_1^\alpha, g_2^\alpha, w_2^\alpha\}$, we know that $h_1^x(p) = b_2^\alpha$, $h_2^x(g_1^\alpha) = w_1^\alpha$, $h_2^x(g_2^\alpha) = g_1^\alpha, h_2^x(w_2^\alpha) = g_2^\alpha$. Since both of $h_1^x(p)$ and $h_2^x(p)$ are in the same gadget with p, there exists a Manhattan path connecting p and $h_1^x(p)$, as well as

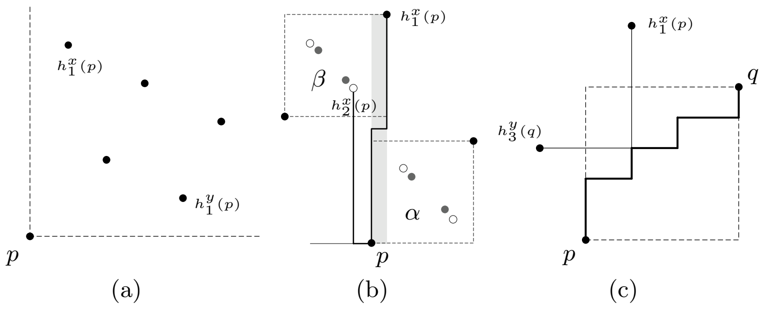

Figure 20: Figure (a) gives an example of $h_1^x(p)$ and $h_1^y(p)$. Figure (b) shows a Manhattan path connecting p and $h_1^x(p)$ or $h_2^x(p)$. Figure (c) shows that the Manhattan path connecting p and q can be obtained by combining two different Manhattan paths together.

p and $h_2^x(p)$ (Property 3). □

In the following analysis, let L_F be the overall length of E_F and L_S the overall length of all the strips, where the length of a horizontal (vertical) strip is the length of the horizontal (vertical) side of the strip. From the construction described above, it is easy to observe that all of these quantities above can be bounded by a polynomial of n and m.

THEOREM 2. *Let*

$$Q := L_F + L_S + \sum \text{cost}_{\min} - 10n_S + 1.$$

Then ψ is satisfiable if and only if there exists a Manhattan network G on T such that $L(G) \leq Q$.

PROOF. First of all, we prove that if ψ is satisfiable, then we can construct a Manhattan network G on T satisfying $L(G) \leq Q$.

We know that for any two points having the same x-coordinate (or y-coordinate), a horizontal line segment (or a vertical line segment) must exist in any Manhattan network. So at the first step we add these line segments into the network, and these line segments compose the set E_F with overall length L_F. Secondly, according to a satisfying assignment of ψ, denoted by π, we construct the nice strip path collection E_S. Then for each gadget, construct the minimum Manhattan network with respect to the assignment π. Since for each gadget, distance 2ϵ is enough for connecting w_1 and g_1, as well as w_2 and g_2, the overall length of line segments added in the third part is at most

$$\begin{aligned} &\sum(\text{cost}_{\min} + 2\epsilon) - 10n_S \\ =&\sum \text{cost}_{\min} - 10n_S + 2\epsilon \cdot n_G \\ \leq&\sum \text{cost}_{\min} - 10n_S + 1 \end{aligned}$$

By definition, the potential cost of each strip S is added to exactly one of its adjacent gadgets. Thus the sum of potential cost used in the network is $10n_S$, and such value has to be subtracted when calculating the minimum length of the network. By Corollary 1 and Corollary 2, it is easy to see that the formula above can be computed in polynomial time.

So we only need to show that

$$L(E_F \cup E_S) = L_F + L_S.$$

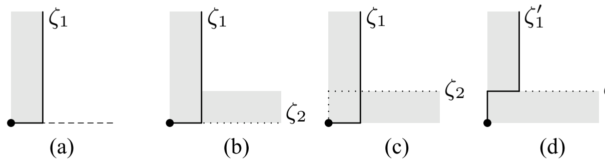

Figure 21: Modifications on the nice strip path collection. The solid line segments represent the strip path ζ_1 or ζ_1'. The dotted line segments represent the strip path ζ_2. The dashed line segment represents a line segment in E_F.

Such equation is satisfied if the switch segment of each strip path falls in E_F, or is shared with another strip path. However, arbitrary union of three mentioned sets does not satisfy this property and we need to modify the generated network. Without loss of generality, we consider the vertical strip and the switch segment is on the bottom. Figure 21 shows all the cases of the switch segment. In Figure 21(a), no strip goes along the bottom side of the gadget, and the switch segment s is shared with a line segment in E_F, whereas in Figure 21(b), the switch segment s of ζ_1 is shared with the strip path ζ_2 of another horizontal strip. In Figure 21(c), the switch segment of ζ_1, and ζ_2, are neither in E_F, nor lying on the other strip path. However, in such case we can modify ζ_1 and let ζ_1', as shown in Figure 21(d), be the new strip path. The above operations are repeated until no such strip path exists. Finally we obtain

$$L(E_F \cup E_S) = L_F + L_S.$$

From the construction of the network, the properties of Theorem 1 are satisfied. Thus the resulting network is a Manhattan network, and the overall length is not greater than

$$L_F + L_S + \sum \text{cost}_{\min} - 10n_S + 1 = Q.$$

On the other hand, we shall prove that if ψ is unsatisfiable, then any network G on the point set T must have $L(G) > Q$. The proof is by contradiction. For the unsatisfiable boolean formula ψ, suppose there exists Manhattan network G satisfying $L(G) \leq Q$. By Theorem 1, G has a strip path collection E_S. Let

$$E_C := G \backslash (E_F \cup E_S).$$

Then we modify the strip paths in E_S. As shown in Figure

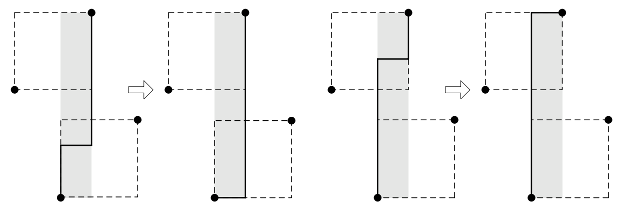

Figure 22: The replacement of strip paths. The strip path, which does not switch in the upper gadget, is replaced by the one representing the assignment 0. Otherwise such a strip path is replaced by the one representing the assignment 1.

22, for a strip path in E_S switching in the middle, we replace it by a strip path which lies on the boundary of the strip. If the strip path does not switch in the upper gadget, we replace it by the path representing value 0. Otherwise such strip path is replaced by the path representing 1.

After these operations, we obtain a nice strip path collection and let π be the corresponding assignment. Again we use the same method, as described before, to make sure that the switch segment of the strip path falls in E_F, or is shared with another strip path. Let the resulting strip path collection be E'_S. We have $L(E_F \cup E'_S) = L_F + L_S$.

Let the network $G' := E_F \cup E'_S \cup E_C$. It is easy to check that G' satisfies Property (1) and (2) in Theorem 1. By the method of modifying E_S, we know that G' also satisfies Property (3). Therefore G' is a Manhattan network, and

$$\begin{aligned} L(G) &= L(E_F \cup E_S \cup E_C) \\ &= L(E_F \cup E_S) + L(E_C) \\ &\geq L_F + L_S + L(E_C) \\ &\geq L(G'). \end{aligned}$$

So the rest is to prove $L(G') > Q$. Let

$$E'_C = G' \backslash (E_F \cup E'_S).$$

Since ψ is unsatisfiable, for at least one gadget the value of cost function exceeds $\text{cost}_{\min}$ by at least 2 under the assignment π. Thus

$$L(E'_C) \geq \sum \text{cost}_{\min} - 10n_S + 2.$$

As a consequence, we have

$$\begin{aligned} L(G') &= L(E_F \cup E'_S) + L(E'_C) \\ &\geq L_F + L_S + \sum \text{cost}_{\min} - 10n_S + 2 \\ &> Q. \end{aligned}$$

Therefore $L(G) \geq L(G') > Q$, which leads to a contradiction. □

Combining Theorem 2 and the fact that 3-SAT is NP-complete, we obtain the main result of this paper.

THEOREM 3. *Minimum Manhattan Network problem is strongly NP-complete, and there does not exist FPTAS algorithms for this problem unless $P = NP$.*

5. ACKNOWLEDGMENTS

Research by Francis Y. L. Chin was supported by GRF, HKU 7113/07E. Zeyu Guo was supported by Hui-Chun Chin and Tsung-Dao Lee Chinese Undergraduate Research Endowment. Research by He Sun was supported by Shanghai Leading Academic Discipline Project No. B114 and NSFC No. 60496321. The authors of this paper thank the referees for their constructive comments and Hong Zhu for useful discussions.

6. REFERENCES

[1] M. Benkert, A. Wolff, and F. Widmann. The minimum Manhattan network problem: a fast factor-3 approximation. In *Proceedings of the 8th Japanese Conference on Discrete and Computational Geometry*, pages 16–28, 2005.

[2] M. Benkert, A. Wolff, F. Widmann, and T. Shirabe. The minimum Manhattan network problem: approximations and exact solutions. *Computational Geometry: Theory and Applications*, 35:188–208, 2006.

[3] V. Chepoi, K. Nouioua, and Y. Vaxès. A rounding algorithm for approximating minimum Manhattan networks. *Theoretical Computer Science*, 390:56–69, 2008.

[4] J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Approximating a minimum Manhattan network. *Nordic Journal of Computing*, 8:219–232, 2001.

[5] Z. Guo, H. Sun, and H. Zhu. A fast 2-approximation algorithm for the minimum Manhattan network problem. In *Proceedings of 4th International Conference on Algorithmic Aspects in Information Management*, pages 212–223, 2008.

[6] Z. Guo, H. Sun, and H. Zhu. Greedy construction of 2-approximation minimum Manhattan network. In *Proceedings of the 19th International Symposium on Algorithms and Computation*, pages 4–15, 2008.

[7] R. Kato, K. Imai, and T. Asano. An improved algorithm for the minimum Manhattan network problem. In *Proceedings of the 13th International Symposium on Algorithms and Computation*, pages 344–356, 2002.

[8] K. Nouioua. Enveloppes de Pareto et Réseaux de Manhattan: Caractérisations et algorithmes. *Ph.D. thesis, Université de la Méditerranée*, 2005.

[9] S. Seibert and W. Unger. A 1.5-approximation of the minimal Manhattan network problem. In *Proceedings of the 16th International Symposium on Algorithms and Computation*, pages 246–255, 2005.

[10] M. Zachariasen. A catalog of Hanan grid problems. *Networks*, 38:76–83, 2001.