Minimum Manhattan Network is NP-Complete

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

Zeyu Guo
School of Computer Science
Fudan University
gzy@fudan.edu.cn

He Sun†
School of Computer Science
Fudan University
sunhe@fudan.edu.cn

March 22, 2010

Abstract

Given a set T of n points in $\mathbb{R}^2$, a Manhattan network on T is a graph G with the property that for each pair of points in T, G contains a rectilinear path between them of length equal to their distance in the L_1-metric. The minimum Manhattan network problem is to find a Manhattan network of minimum length, i.e. minimizing the total length of the line segments in the network.

In this paper, we prove that the decision version of the MMN problem is strongly **NP**-complete, using a 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 a 3-CNF formula.

Keywords: Minimum Manhattan networks, 3-SAT, NP-completeness

*The research was in parts supported by a GRF grant in Hong Kong, HKU 7116/08E.
†Corresponding author

1 Introduction

1.1 Problem Description

A *rectilinear path* connecting two points in the plane is a path consisting of only horizontal and vertical line segments. A *Manhattan path* between p and q is a rectilinear path with its length equal to $|p.x - q.x| + |p.y - q.y|$, *i.e.*, the L_1-distance between p and q.

Given a set T of n points in $\mathbb{R}^2$, a Manhattan network on T is a graph $G = (V, E)$ with the property that all the edges in E are vertical or horizontal line segments connecting points in $V \supseteq T$ and for all $p, q \in T$, the graph contains a path having the length exactly the L_1-distance between p and q. The length of a network G, denoted by $L(G)$, is the total length of 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 $L(G)$.

The MMN problem is closely connected to planar t-spanners. For a number $t \geq 1$, a planar graph G is said to be a t-spanner of a point set 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 Euclidean 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-metric [3, 5].

1.2 Historical Review

Motivated by a number of applications in city planning, network layouts, distributed algorithms and VLSI circuit design, the MMN problem was first introduced in 1999 by Gudmundsson et al. [5]. 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 the MMN problem is **NP**-hard; (2) whether or not a PTAS exists for the MMN problem; and (3) whether or not a 2-approximation algorithm exists for the MMN problem.

After [5], much research was devoted to finding approximation algorithms for the MMN problem. Most combinatorial constructions [1, 2, 6, 11] rely on the decomposition of the input, by partitioning the plane into several blocks (ortho-convex regions) that can be solved independently. Kato et al. [8] 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 $T \times T$, is sufficient for determining whether a given network is a Manhattan network. Following this idea, 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, Chepoi et al. [3] used the notion *Pareto Envelope* and a nice strip-staircase decomposition to divide the plane into several regions, which can be studied individually. Based on this idea, they proposed a 2-approximation rounding algorithm by solving the linear programming relaxation of the MIP. In Nouioua's Ph.D. thesis [10], he presented a primal-dual based algorithm that yields a 2-approximation and runs in $O(n \log n)$ time. Later, Guo et al. [6] showed 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 for the quadrangle inequality. Furthermore, Guo et al. [7] presented a 2-approximation algorithm for constructing a Manhattan network in $O(n \log n)$ time. Compared with the previously best known combinatorial construction [6], Guo et al. [7] used a simple greedy strategy to construct a 2-approximate MMN. Seibert et al. [11] proposed a 1.5-approximation algorithm. However, their proof may be incorrect [3]. Muñoz et al. [9] gave an **NP**-hardness proof of this problem in three dimensions and showed that there is no polynomial-time approximation algorithm with a ratio better than $1 + 2 \cdot 10^{-5}$ under the assumption $\mathbf{P} \neq \mathbf{NP}$.

Despite the wealth of publications in this area it has been an open question since 1999 whether the MMN problem is **NP**-hard or not.

1.3 Our Approach and Results

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

The construction of the gadgets and the reduction rely on the following ideas: (1) A horizontal or vertical line segment always exists between two points having the same x- or y-coordinate, see Fig. 1(a). (2) There are many ways of connecting two points with different x- and y-coordinates. However, for a set of point pairs, forming strips, we are only interested in two different connections which, roughly speaking, use one

horizontal and one vertical line segment. These two ways of connecting two points in a strip can be shown optimal and will be used to represent the assignment of 0 or 1 to a boolean variable, see Fig. 1(b). (3) There are several ways of connecting the points within a gadget and these form the bases of different gadgets with various functions, see Fig. 1(c).

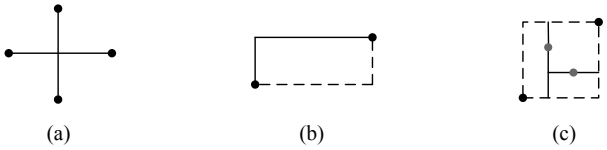


Figure 1: Intuition behind our construction. (a) The line segments with endpoints having the same x- or y-coordinate compose a set E_F; (b) The set E_S consists of strip paths for every strip; (c) The solid line segments within each gadget compose a set E_C.

Our reduction generates a point set T from any 3-CNF 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.* $G = (V, E_F \cup E_S \cup E_C)$. The set E_F consists of all line segments with endpoints $p, q \in T$ based on the first idea, where $p.x = q.x$ or $p.y = q.y$. Basically E_F contains those line segments whose endpoints have the same x- or y-coordinate and form the basic structure of all gadgets; E_S is a set of strip paths based on the second idea, and each path, connecting two points of different coordinates, usually consists of one vertical line segment and one horizontal line segment; 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. The set E_C are line segments connecting one or two isolated points in each gadget to other points in the same gadget based on the third idea. Let ℓ_α be the total length 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 a relationship among these assignments in each gadget. The total length of E_C can be computed by $\sum_\alpha \ell_\alpha$ and reaches minimum if each ℓ_α achieves the minimum value. Unfortunately the values of ℓ are not independent, and 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 to be the sum of ℓ_α and the potential cost of all strips associated with the gadget α. We will show that the value of E_C is bounded

by a certain value if and only if the assignments of all strips (variables) give a minimum cost_α for each gadget α. Furthermore, we show that cost_α achieves a 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, we prove that the MMN problem is strongly **NP**-complete and there does not exist an FPTAS for this problem unless $\mathbf{P} = \mathbf{NP}$. This also answers the first problem stated by Gudmundsson et al. [5], which has been open for more than ten years.

This paper is structured as follows: Sect. 2 gives the basic notations and concepts, followed by our reduction in Sect. 3. Sect. 4 is devoted to the correctness proof of the reduction. We end this paper with some open problems in Sect. 5.

2 Preliminaries

Given $p, q \in \mathbb{R}^2$, let $R(p, q)$ be the smallest axis-aligned closed rectangle that contains p and q. The area $B_V(p, q)$ is defined as the vertical closed region bounded by p, q, *i.e.*

$$B_V(p, q) := \Big\{(x, y) \mid \min\{p.x, q.x\} \leq x \leq \max\{p.x, q.x\}, y \in \mathbb{R}\Big\},$$

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

$$B_H(p, q) := \Big\{(x, y) \mid \min\{p.y, q.y\} \leq y \leq \max\{p.y, q.y\}, x \in \mathbb{R}\Big\}.$$

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 $\{(x, y)|x = p.x, y \leq p.y\}$ and $\{(x, y)|x = q.x, y \geq q.y\}$ (represented by the dashed lines in Fig. 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 $|p.x - q.x|$. A *horizontal strip* is defined similarly, see Fig. 2. For any strip $R(p, q)$, we call a Manhattan path between p and q a *strip path* of $R(p, q)$. Moreover, for each strip we choose one Manhattan path and these paths compose a set E_S, called *a strip path collection*. A 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 reduction, each strip is used to express an assignment of a boolean variable, which is indicated by which side of the strip the path (network) includes. Specifically, by

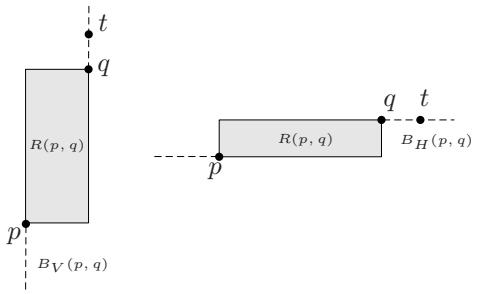


Figure 2: The rectangle is a vertical/horizontal strip $R(p, q)$. Any point in T within $B_V(p, q)$ or $B_H(p, q)$ can only be placed on the dashed lines, *e.g.*, point t.

definition of vertical strips, we can assume that the strip path of $R(p, q)$ will not contain any vertical line segment whose x-coordinate is not equal to $p.x$ or $q.x$ [12], 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 a variable v in ψ. This nice property of a strip path is also used to transport the value of v from one end into the other. As shown in Fig. 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 Sect. 4, we show that all the other strip paths can be reduced to the nice ones for the point set considered in this paper.

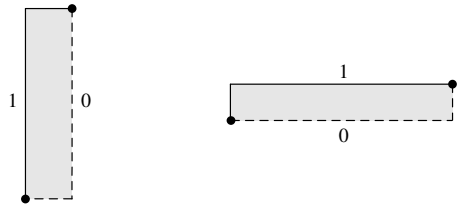


Figure 3: For each strip, two types of Manhattan paths are used to represent the assignment of a variable v. The solid line segments represent $v = 1$, and the dashed line segments represent $v = 0$.

3 Reduction

3.1 Reduction Overview

We first give a high-level overview of our reduction. The main task is to convert any given 3-CNF 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 an MMN G on T is bounded by a polynomial-time

computable value.

As shown in Fig. 4, each line segment, denoting one strip, is associated with a literal (with vertical lines for vertical strips and horizontal lines for horizontal strips), each circled node represents one of the gadgets, and each dotted-line square represents one clause in ψ. Specifically, the two adjacent vertical lines numbered by $2i-1, 2i, 1 \leq i \leq n$, which might be segmented by `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 the boolean values represented by the connections in the network can be flipped or maintained. These vertical line segments (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 a `SPLITTER` gadget, to a `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. A `NEGATOR` gadget together with a `CLAUSE` gadget is to ensure that the connection will achieve the lowest length if the clause is satisfiable. In other words, the length of the Manhattan network on T will be bounded by a certain value if and only if there is an assignment satisfying every clause. In order to construct the desired network, the strips in our construction have two different widths—standard width 20 and narrow width 10, and an `ADAPTOR` gadget is for connecting two strips with different widths.

3.2 Gadget Design

Our reduction relies on six different gadgets: `NEGATOR, TURN, SPLITTER, CLAUSE, ADAPTOR`, and `DUMMY`. The gadgets are placed in such a way that there exists a strip, either horizontal or vertical, between two gadgets, see Fig. 4. Generally speaking, each gadget 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, see Fig. 5. The positions of points inside each gadget are as follows: $b_1.x < w_1.x < g_1.x \leq g_2.x < w_2.x < b_2.x$, $b_1.y <$

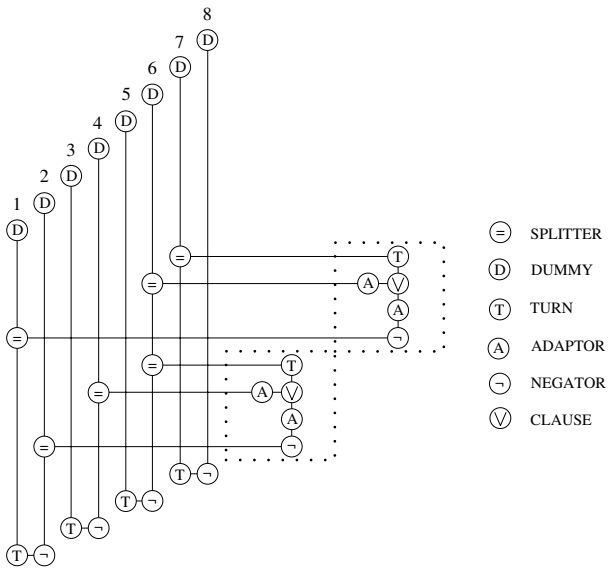


Figure 4: A schematic view of the construction. Each line segment represents a strip, and each circled node represents a gadget. On the left side of this figure, vertical strips are used to represent the assignments of n variables in ψ, and on the right side of the figure, each dotted-line square, consisting of five gadgets, corresponds to a clause. This graph corresponds to the input 3-CNF formula $\psi = (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$.

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

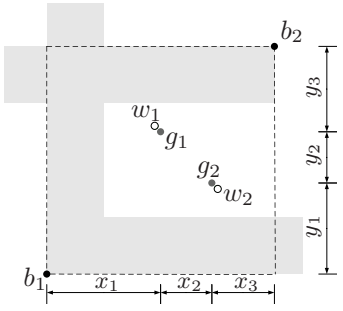


Figure 5: The configuration of gadgets. Generally speaking, each gadget consists of two black points, two white points and two grey ones. In the degenerated case, two grey points may coincide and the gadget has only five points. The grey regions represent three strips.

The two grey points are very close to the white points with distance ϵ under the L_1-metric, $\epsilon \leq \frac{1}{2n_G}$, where n_G is the number of gadgets and will be determined later. Notice that two grey points may coincide in the degenerated case. We assume that the length of the connection between w_1 and g_1, and similarly w_2 and g_2, is very small, and they can be ignored as long as the total length of an MMN is bounded by a certain

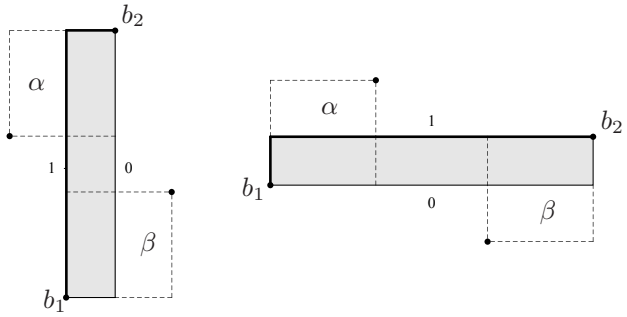


Figure 6: The Manhattan path representing 1 is partially inside the gadget α and on the boundary of the gadget β.

value. In the following discussion of gadgets, the gadgets are placed in such a way that b_1 of a gadget α, expressed by b_1^α, and b_2 of another gadget β will form a strip $R(b_1^\alpha, b_2^\beta)$. Note that b_1 will always be located at the lower left corner of the strip and b_2 the upper right corner. Depending on their relative positions, horizontal or vertical strips will be formed according to the schematic diagram of Fig. 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, see Fig. 6. It is always advantageous if the strip path lies on the boundary of the strip, as such a path is inside one gadget and will shorten the connection length within the gadget. That is the intuitive reason (formal proof in Sect. 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 boundary of both gadgets, which results in larger connection length for both gadgets.

Let B be a big-parallel rectangle enclosing all the gadgets and strips. Assume the four boundary edges ∂B of B are on the lines $x = x_1, x = x_2, y = y_1, y = y_2$ respectively ($x_1 < x_2, y_1 < y_2$). Initially, let T_0 be the corner point set of B, *i.e.*, $T_0 := \{(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)\}$, and let $T := T_0$. 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, see Fig. 7. When there is no strip along some side of a gadget, one point of ∂B will 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 a strip. For example, in Fig. 7, there is no strip on the right side of the gadget, therefore a point u is placed on the bottom edge of ∂B with its x-coordinate same as $b_2.x$. Similar construction is applied

for the other three sides. By adding these points, some line segments, actually in E_F and represented by the solid line segments in Fig. 7, 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.

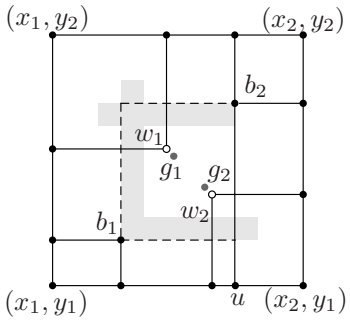


Figure 7: The extra points of ∂B with respect to one gadget. For each gadget, we add eight points in the resultant network. Based on the relative positions among different gadgets we add a point u in the resulting network.

In the following, we describe the `NEGATOR` gadget in detail and outline the other gadgets. Since the distance between g_1 and w_1, and similarly g_2 and w_2, is very small, we 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 a 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 the boundary of the gadget. This will make the analysis difficult if each gadget is studied in isolation, as each strip path always runs 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 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 is included in the cost calculation if the strip path is inside the gadget, *i.e.* $v = 0$ when the strip is on the left or the top of the gadget, or $v = 1$ when the strip is on the right or the bottom of the gadget. Informally, the potential cost of each strip will be added into the cost evaluation of a gadget α if and only if the strip path is in

the interior of α.

As shown in Fig. 8, 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 variables. The four possible connections are indicated by the dashed line segments as shown from Fig. 9 to Fig. 12.

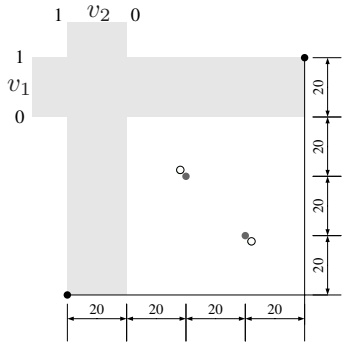


Figure 8: NEGATOR

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

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

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 will show that for this case, the length of connection in other gadgets is not lowest. An assignment of a strip always affects two gadgets and the same strip always lies at different sides of the two gadgets, at the top side of one and the 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. A potential cost of 10 is added to exactly one of two gadgets adjacent to a strip when evaluating the cost function. For example, in Fig. 6, a potential cost 10 will be added to the cost of α if we choose the Manhattan path representing 1; otherwise the potential cost is counted when calculating the cost function of β. Since the number of

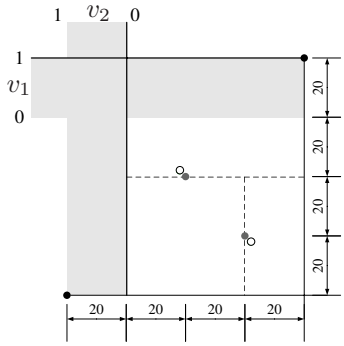


Figure 9: $v_1 = 1, v_2 = 0, \ell = 100$

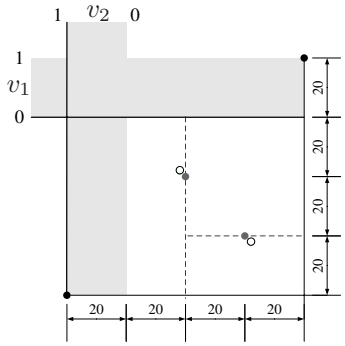


Figure 10: $v_1 = 0, v_2 = 1, \ell = 100$

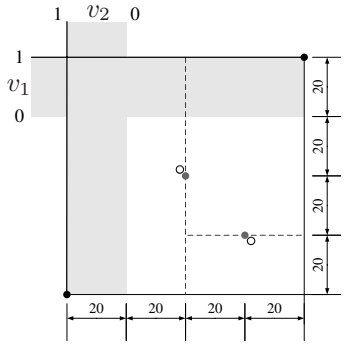


Figure 11: $v_1 = 1, v_2 = 1, \ell = 120$

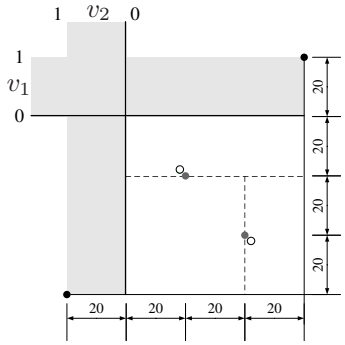


Figure 12: $v_1 = 0, v_2 = 0, \ell = 100$

strips, denoted by n_S, is fixed for the input formula ψ, the sum of added potential costs $10n_S$ will be subtracted and this way of handling the potential costs will not affect the final length of the network.

By including this potential cost, the adjusted costs for the four cases are shown in Table 1, where the lowest cost is achieved when $v_1 = \neg v_2$. Similar analysis is applied for the other gadgets, with the total length, potential cost and adjusted cost of each assignment listed in the tables. The correctness of all the tables was verified with a computer assistance, by enumerating all the connections of points within the gadget and choosing the lowest one.

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 Fig. 13. The positions of the white and grey points in the gadget guarantee that the cost function achieves minimum if and only if all of the three strips get the same boolean value (Fig. 13, Table 2).

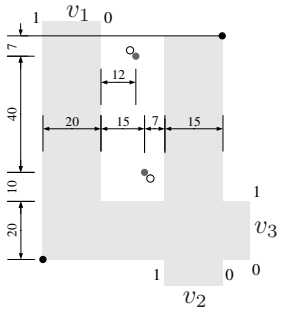


Figure 13: SPLITTER

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

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

Since originally the assignment of each variable is represented by vertical strips, the TURN gadget is used to change the direction of an information flow so that a horizontal strip can represent the same assignment with the crossing vertical strip, and cost_{min} in the TURN gadget is obtained if and only if these two crossing strips express the same assignment (Fig. 14 and Fig. 15, Table 3).

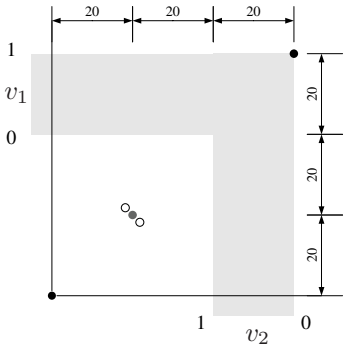


Figure 14: TURN(a)

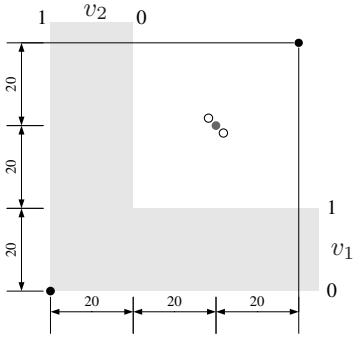


Figure 15: TURN(b)

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

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

Three strips in a CLAUSE gadget (Fig. 16) represent 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 a `CLAUSE` gadget in seven out of eight assignments of three variables (Table 4). Note that the cost function will not achieve minimum only for the case $v_1 = 0, v_2 = 1, v_3 = 0$. Since the strip representing v_2 is connected to the left vertical strip through a `NEGATOR` gadget, $\text{cost}_{\min}$ for the `CLAUSE` gadget can be achieved if and only if the corresponding clause is satisfied, *i.e.* as long as not all the values of the three variables are 0.

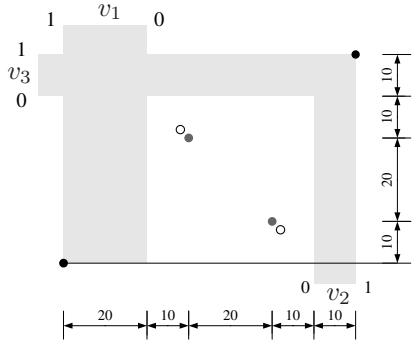


Figure 16: `CLAUSE`

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

In order to achieve this goal, the width of two out of the three strips in a `CLAUSE` gadget is set to 10 which is smaller than the standard width of 20. An `ADAPTOR` gadget (Fig. 17 and Fig. 18, 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.

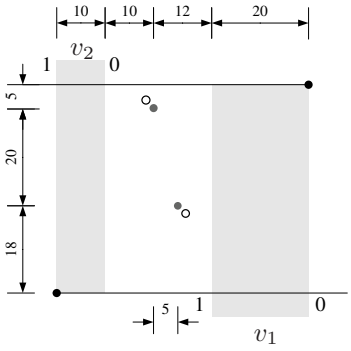


Figure 17: `ADAPTOR`(a)

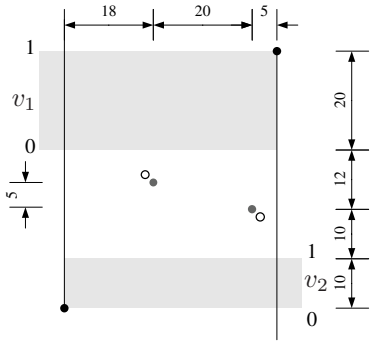
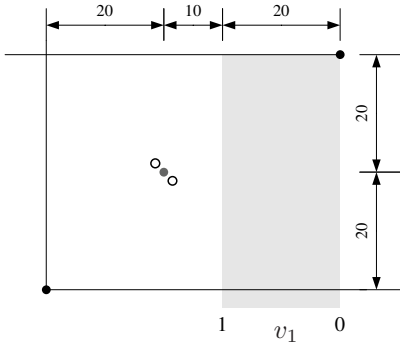


Figure 18: `ADAPTOR`(b)

A `DUMMY` gadget (Fig. 19, Table 6) is located at the end of each vertical strip associated with each literal.

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

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



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

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

Figure 19: DUMMY

3.5 Putting Them Together

According to Fig. 4, we combine different gadgets to form the desired network. For instance, a SPLITTER gadget is connected to gadgets DUMMY, TURN, and NEGATOR. So we let a DUMMY gadget and a SPLITTER gadget share a common vertical strip, a TURN gadget and a SPLITTER gadget share a common vertical strip, whereas a horizontal strip of a SPLITTER gadget is connected to a NEGATOR gadget, see Fig. 20. Similar approach is applied for connecting other gadgets and obtaining the point set T.

In addition, if a gadget α is located to the left of a gadget β in Fig. 4, then we put all the points in α to the left of the points in β, *i.e.* $b_2^\alpha.x < b_1^\beta.x$. Similarly, if a gadget α is located above a gadget β in Fig. 4, then we put all the points in α above the points in β, *i.e.* $b_1^\alpha.y > b_2^\beta.y$.

As described before, we only need to consider the optimal network connection of each gadget without worrying about the global network length. Actually we shall prove in Sect. 4 that ψ is satisfiable if and only if $\text{cost}_{\min}$ is 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. *The minimum cost value* $\text{cost}_{\min}$ *is obtained*

- *for* **any** *assignment of the strip in the* DUMMY *gadget.*

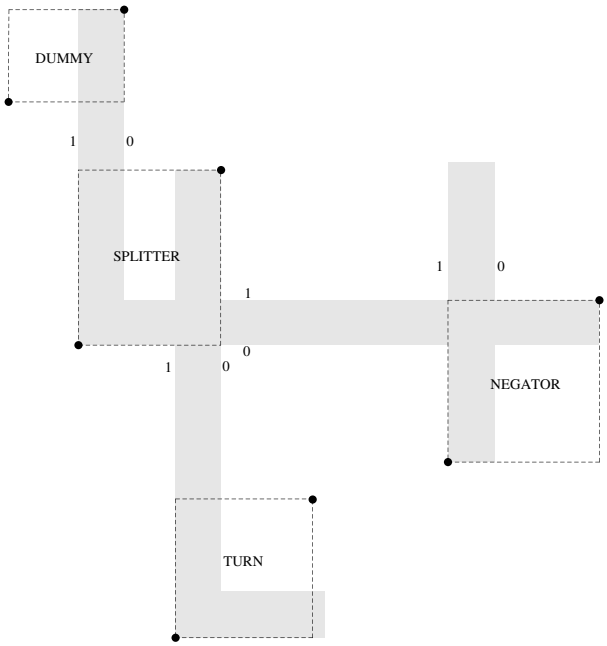


Figure 20: The method of combining different gadgets. Two adjacent gadgets always share a common strip.

- *if the crossing strips represent* **different** *boolean values in the* NEGATOR *gadget.*
- *if all the crossing strips represent the* **same** *boolean 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 Fig. 4, we have the following facts.

Lemma 2. *For any 3-CNF 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.*

Proof. As shown in Fig. 4, each dotted-line square, consisting of one CLAUSE gadget, two ADAPTOR gadgets, one TURN gadget and one NEGATOR gadget, represents a clause of ψ. On the left side of Fig. 4, two DUMMY gadgets, one TURN gadget and one NEGATOR gadget

are used to express the assignment of each variable. Three `SPLITTER` gadgets are used with respect to three literals in each clause. □

Corollary 1. *In the resulting instance reduced from ψ with n variables and m clauses, the number of gadgets $n_G = 8m + 4n$, and the number of strips $n_S = 10m + 3n$.*

Proof. We divide the gadgets into three categories:

- Each `DUMMY` gadget is associated with one strip.
- For the gadgets `ADAPTOR`, `TURN` and `NEGATOR`, each gadget is associated with two strips.
- For the `CLAUSE` and `SPLITTER` gadgets, each gadget is associated with three strips.

Combing this fact with Lemma 2, the number of strips $n_S = 10m + 3n$.

The number of gadgets follows from Lemma 2 directly. □

Corollary 2. *For the resulting instance reduced from ψ with n variables and m clauses, $\sum \text{cost}_{\min} = 240n + 641m$.*

Proof. Sum up the $\text{cost}_{\min}$ values for all the gadgets according to Table 1-6. □

Recall that all the line segment connecting the points with the same x- or y-coordinate compose a set E_F. A set E_S consists of strip paths and E_C contains the line segments within each gadget. Now we prove that G, consisting of E_F, E_S and E_C, is indeed a Manhattan network.

Define $\mathcal{Q}_k(p)$ as 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 point exist, choose the point whose y-coordinate is closest to $p.y$). Similarly, we can give the definition of $h_k^y(p)$. Fig. 21(a) shows an example of $h_1^x(p)$ and $h_1^y(p)$. For convenience, points in T_0 are excluded from the domains of h_k^x and h_k^y so that $h_k^x(p)$ and $h_k^y(p)$ always exist.

The following lemma, first presented by Gudmundsson et al. [5], gave an efficient way of verifying whether a network is a Manhattan network. For completeness, we give a proof of this lemma for the network we consider.

Lemma 3. *Given a network G, if for any point $p \in T \backslash T_0$, there exist a Manhattan path 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 a Manhattan path between p and q.

First, we discuss the case that $p \in T_0$ or $q \in T_0$. Without loss of generality, assume $p = (x_1, y_1)$, see Fig. 7. Since $T \cap R(p, q) \backslash \{p, q\} = \emptyset$, $q \notin T_0$. We have $h_3^x(q) = p$ and p, q are connected by a Manhattan path.

If both $p \notin T_0$ and $q \notin T_0$, assume $p.x \leq q.x, p.y \leq q.y$ and the other cases can be proven similarly. In such a case, the path between p and $h_1^x(p)$ and the path between q and $h_3^y(q)$ always intersect, and as shown in Fig. 21(b), 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. □

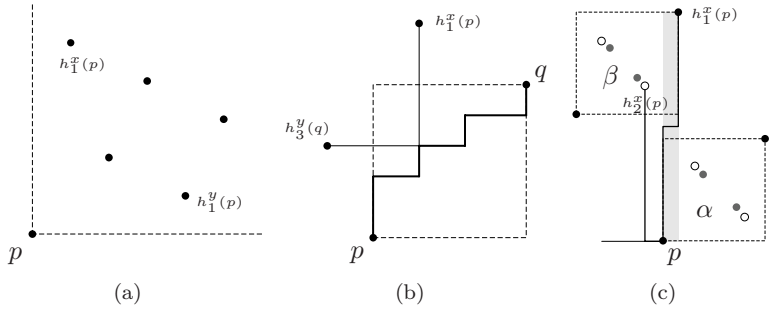


Figure 21: (a) an example of $h_1^x(p)$ and $h_1^y(p)$. (b) a Manhattan path between p and q can be obtained by combining two different Manhattan paths together. (c) a Manhattan path connecting p and $h_1^x(p)$ or $h_2^x(p)$.

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; and (3) G contains a Manhattan subnetwork on the vertex set of each gadget.*

Proof. It is easy to see that a Manhattan network satisfies 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 pair of points $p, h_1^x(p)$, as well as $p, h_2^x(p)$. If there exists a point $u \in T$ such that $u.x = p.x, u.y > p.y$, and there is no point in 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). We can verify that if p is the point w_1 or b_2 of a gadget, then such a point u exists. Similarly, u also exists if p is a point in $\partial B \backslash T_0$ but is not on the top boundary. Therefore in these cases, there exist a Manhattan path between p and $h_1^x(p)$, as well as p and $h_2^x(p)$.

For the point p lying on the top boundary of ∂B, the top horizontal line segment $(y = y_2) \cap \partial B$ contains a Manhattan path 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 $\{b_1, g_1, g_2, w_2\}$ in any gadget α, where there is no point $u \in T$ satisfying $u.x = p.x, u.y > p.y$.

If $p = b_1$ of a gadget α, denoted by b_1^α, then there exists a gadget β such that $R(b_1^\alpha, b_2^\beta)$ is a vertical strip. As shown in Fig. 21(c), $h_1^x(p) = b_2^\beta$ and $h_2^x(p) = w_2^\beta$. Points p and $h_1^x(p)$ are connected by a Manhattan path in the strip path collection (Property 2), whereas a 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, $p \in \{g_1^\alpha, g_2^\alpha, w_2^\alpha\}$ and 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 p and $h_2^x(p)$ (Property 3). □

Let L_F be the overall length of E_F, and L_S be 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, all of these quantities are bounded by a polynomial in 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 have a Manhattan network G on T satisfying $L(G) \leq Q$.

We know that for any two points having the same x- or y-coordinate, a horizontal or 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. Thirdly, we add the line segments within each gadget according to π. Since for each gadget, the length of line segments used for connecting w_i and g_i, $i \in \{1, 2\}$, is at most ϵ, the overall length of line segments added in the third step 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 is added to exactly one of its adjacent gadgets. Thus the sum of the potential costs used in the network is $10n_S$, and this value is subtracted when calculating the length of the network. By Corollary 1 and Corollary 2, the above formula can be computed in polynomial-time.

Next we show that $L(E_F \cup E_S) = L_F + L_S$. This equation holds if the switch segment of each strip path falls in E_F, or is shared with another strip path. However, arbitrary union of E_F, E_S and E_C does not satisfy this property and we need to modify the generated network. Without loss of generality, we consider a vertical strip where the switch segment is at the bottom. Fig. 22 shows all the cases of the switch segment, and ζ_1, ζ_2 are two strip paths for the vertical and horizontal strips. In Fig. 22(a), no strip goes along the bottom side of the gadget, and the switch segment is shared with a line segment in E_F, whereas in Fig. 22(b), the switch segment of ζ_1 is shared with the strip path ζ_2 of another horizontal strip. In Fig. 22(c), the switch segment of ζ_1 (ζ_2) is neither in E_F, nor lying on the other strip path. However, in this case we modify ζ_1 and let ζ_1' be the new strip path, see Fig. 22(d). We repeat these operations until no such strip path exists. Finally we get $L(E_F \cup E_S) = L_F + L_S$.

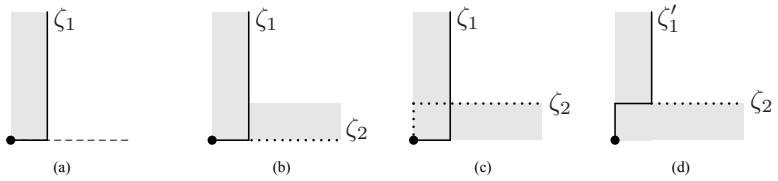


Figure 22: Modifications on the nice strip path collection. The solid line segments represent the strip path ζ_1 or ζ_1'. The dotted line segments represent the strip path ζ_2. The dashed line segment represents a line segment in E_F.

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 prove that if ψ is unsatisfiable, then any network G on T must have $L(G) > Q$. 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 Fig. 24, 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 a strip path is replaced by the path representing 1.

It can be observed that our way of replacing a strip path ζ by a new path ζ' changes the local connection of a gadget only when the switch segment of ζ lies in this gadget. Consider such a strip path ζ whose switch segment is in a gadget. Without loss of generality, we assume the strip is vertical and the gadget is at its lower end, as shown in Fig. 23. Notice that the only Manhattan paths within the gadget that may be affected are those connected to b_1. For these connections, the path ζ' is used instead to compose new Manhattan paths. So Property (3) in Theorem 1 is maintained when performing the replacement operations.

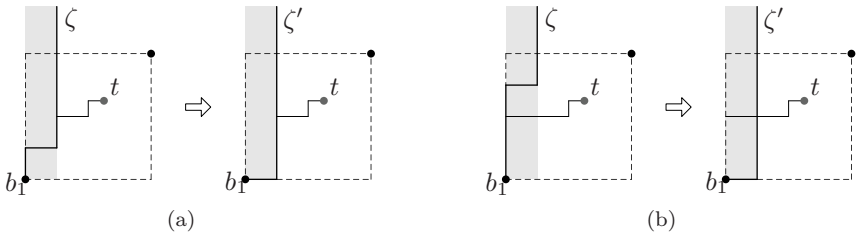


Figure 23: After modifying a Manhattan path ζ, only the paths connecting b_1 will be affected, *e.g.* the path between b_1 and t. However, a path ζ' is used so that these connections are actually preserved.

After these operations, we have 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 each strip path falls in E_F, or is shared with another strip path. This process maintains Property (3). Let the resulting strip path collection be E'_S. We have $L(E_F \cup E'_S) = L_F + L_S$.

Let the network $G' := E_F \cup E'_S \cup E_C$. By the method of modifying E_S, G' satisfies Property (3). Property (1) and (2) also hold since G consists of E_F and E'_S. 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 any assignment. 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 $L(G) \geq L(G') > Q$. □

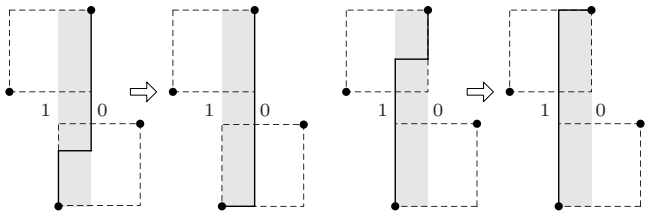


Figure 24: The replacement of strip paths. The strip path, which does not switch in the upper gadget, is replaced by the path representing the assignment 0. Otherwise such a strip path is replaced by the path representing the assignment 1.

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

Theorem 3. *The minimum Manhattan network problem is strongly* **NP***-complete, and there does not exist an FPTAS for this problem unless* $\mathbf{P} = \mathbf{NP}$.

Proof. We have presented a polynomial-time reduction from the 3-SAT problem. Since all numerical parameters used are bounded by a polynomial in n and m, the MMN problem is strongly **NP**-hard. Now we show that it is in **NP**. By [12], there exists a Manhattan network in the Hanan grid of T which contains a polynomial number of edges. Let the network, denoted by G, be a certificate and it can be verified in polynomial-time by computing $L(G)$ and comparing $L(G)$ with Q. □

5 Open problems

Several interesting problems remain open. First of all, it is not known whether a PTAS exists for this problem. Muñoz et al. [9] gave an inapproximation factor in three dimensions. However, both our reduction presented in this paper and the **NP**-completeness proof of the MMN problem in three dimensions [9] can not be applied directly to obtain an inapproximation ratio of the MMN problem in two dimensions.

Another interesting problem is to design approximation algorithms for the MMN problem in higher dimensions. Gudmundsson et al. [4] showed that for any n points in $\mathbb{R}^d$, there is a Manhattan network that consists of only $O(n \log^{d-1} n)$ vertices and edges. Muñoz et al. [9] generalized the notion of critical rectangles and gave the definition of critical cuboids. Even though some preliminary attempts have been shown in those papers, it is unknown how to design constant-factor approximation algorithms in three dimensions.

Acknowledge: We thank the anonymous referees for their valuable comments.

Appendix

The demo of our reduction is available at

`http://www.tcs.fudan.edu.cn/~sun/mmnc.rar`

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. *Comp. Geom.-Theor. Appl.*, 35:188–208, 2006.

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

[4] J. Gudmundsson, O. Klein, C. Knauer, and M. Smid. Small Manhattan networks and algorithmic applications for the Earth mover's distance. In *Proceedings of the 23rd European Workshop on Computional Geometry*, pages 174–177, 2007.

[5] J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Approximating a minimum Manhattan network. *Nord. J. Comput.*, 8:219–232, 2001.

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

[7] 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.

[8] 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.

[9] X. Muñoz, S. Seibert, and W. Unger. The minimal Manhattan network problem in three dimensions. In *Proceedings of the 3rd International Workshop on Algorithms and Computation*, pages 369–380, 2009.

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

[11] 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.

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