Minimum Weight Triangulation is NP-Hard

Wolfgang Mulzer
Department of Computer Science
Princeton University
35 Olden Street
Princeton, NJ 08540, USA
wmulzer@cs.princeton.edu

Günter Rote
Institut für Informatik
Freie Universität Berlin
Takustraße 9
D-14195 Berlin, Germany
rote@inf.fu-berlin.de

ABSTRACT

A triangulation of a planar point set S is a maximal plane straight-line graph with vertex set S. In the *minimum weight triangulation* (MWT) problem, we are looking for a triangulation of a given point set that minimizes the sum of the edge lengths. We prove that the decision version of this problem is NP-hard. We use a reduction from PLANAR-1-IN-3-SAT. The correct working of the gadgets is established with computer assistance, using geometric inclusion and exclusion criteria for MWT edges, such as the diamond test and the LMT-Skeleton heuristic, as well as dynamic programming on polygonal faces.

General Terms

Algorithms, Theory

Keywords

Optimal triangulations, PLANAR-1-IN-3-SAT

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Geometrical problems and computations; G.2.2 [**Graph Theory**]: Graph algorithms

1. INTRODUCTION

Given a set S of points in the euclidean plane, a *triangulation* T of S is a maximal plane straight-line graph with vertex set S. The *weight* of T is defined as the total euclidean length of all edges in T. A triangulation that achieves minimum weight is called a *minimum weight triangulation* (MWT) of S.

The problem of computing a triangulation for a given planar point set S arises naturally in many applications such as stock cutting, finite element analysis, terrain modeling, and numerical approximation. The *minimum-weight* triangulation has attracted the attention of many researchers, mainly due to its natural definition of optimality, and not so much because it would be important for the mentioned applications. We show that it is NP-hard to compute a minimum-weight triangulation of a given planar point set. Our proof uses a polynomial time reduction from PLANAR-1-IN-3-SAT. The input to the PLANAR-1-IN-3-SAT problem consists of a Boolean formula Φ in 3-CNF whose associated graph is planar, and the formula is accepted if there exists an assignment to its variables such that in each clause exactly one literal is satisfied. The *associated graph* of Φ has one vertex v_x for each variable x in Φ and one vertex v_C for each clause C in Φ. There is an edge between v_x and v_C if and only if x or $\neg x$ appears in C. PLANAR-1-IN-3-SAT is NP-complete (see Section 2.2).

1.1 History and previous results

The minimum weight triangulation problem has a long and rich history, dating back to the 1970s. At the end of Garey and Johnson's book from 1979 on NP-completeness [14], there is a list of 12 major problems whose complexity status was open at the time of writing. So far, 9 problems from this list have been resolved by proving NP-hardness or by exhibiting a polynomial-time algorithm, the latest one being the deterministic prime number test (see [18] for a recent status update on the list). With the present paper, only two problems from the original list remain open.

It seems that the MWT problem was first considered by Duppe and Gottschalk [13] who propose the greedy algorithm for solving the MWT problem which always adds the shortest possible edge to the triangulation. Later, Shamos and Hoey [32] suggest using the Delaunay triangulation as a minimum weight triangulation. In 1977, Lloyd [27] provides examples which show that both proposed algorithms usually do not compute the MWT. He also shows that given a set of edges of a planar point set, it is NP-complete to decide whether it contains a triangulation. Later, Gilbert [15] and Klincsek [20] independently show how to compute a minimum weight triangulation of a simple polygon in $O\left(n^3\right)$ time by dynamic programming. There have also been attempts to attack the general problem with dynamic programming techniques. For example, Cheng, Golin, and Tsang [6] use dynamic programming in order to compute a minimum-weight triangulation of a given point set S in $O\left(n^{k+2}\right)$ time if a subgraph of a MWT of S with k connected components is known. Using branch and cut, Kyoda et al. [24] show how to compute MWTs of 100 points. For large point sets, however, mere dynamic programming becomes absolutely infeasible, and hence different ideas are needed.

SCG'06, June 5–7, 2006, Sedona, Arizona, USA.

For special classes of point sets, it is possible to compute the MWT in polynomial time. For example, Anagostou and Corneil [1] give an algorithm to compute the MWT of the vertex set of k nested convex polygons in $O\left(n^{3k+1}\right)$ time. More recently, Hoffmann and Okamoto [17] show how to obtain the MWT of a point set with k inner points in $O\left(6^k n^5 \log n\right)$ time.

In another line of attack, researchers were looking for triangulations that approximate the MWT. The Delaunay triangulation is not a good candidate, since it may be longer by a factor of $\Omega(n)$ [23, 28]. The greedy triangulation approximates the MWT by a factor of $\Theta(\sqrt{n})$ [28, 25, 26]. Plaisted and Hong [29] show how to approximate the MWT up to a factor of $O(\log n)$ in $O\left(n^2 \log n\right)$ time. Levcopoulos and Krznaric [26] introduce quasi-greedy triangulations, which approximate the MWT within a constant factor. Very recently, Remy and Steger [30] discovered an approximation scheme for MWT that runs in quasi-polynomial time: for every fixed ε, it finds a $(1+\varepsilon)$-approximation in $n^{O(\log^8 n)}$ time.

Let us now turn to subgraphs and supergraphs of the MWT. Gilbert [15] shows that the MWT always contains the shortest edge. Yang, Xu, and You [33] extend that result by proving that edges which join mutual nearest neighbors are in the MWT. A larger subgraph of the MWT is identified by Keil [19], who shows that the *β-skeleton*, the graph formed by all edges e such that the two circles of radius $\frac{\beta}{2}|e|$ passing through the endpoints of e are empty, is a subgraph of the MWT for a $\beta \leq \sqrt{2}$. This is improved to $\beta \leq 1.17682$ by Cheng and Xu [8], which is nearly optimal, since there is a lower bound of about 1.154701 [19]. The β-skeleton usually has many connected components. Often, the *LMT-skeleton heuristic* described by Dickerson, Keil, and Montague [10] yields better results. We describe it in the next section.

Approaching the problem from the other direction, Das and Joseph [9] describe the diamond test, which yields a supergraph of the MWT: An edge e can only be contained in the MWT if at least one of the two isosceles triangles with base e and base angles $\pi/8$ is empty. This constant is improved to $\pi/4.6$ by Drysdale, McElfresh, and Snoeyink [11]. The diamond test gives an easy criterion to exclude impossible edges from the MWT. Usually, this eliminates all edges except a set of $O(n)$ remaining candidate edges. (This statement is true for random point sets, with high probability. With bucketing techniques, such a set of $O(n)$ edges can be found in linear expected time [12].)

2. PRELIMINARIES

2.1 The LMT-Skeleton

The *locally-minimum triangulation* (LMT) heuristic was introduced by Dickerson, Keil, and Montague [10]. The idea is that an edge e can only be included in the MWT if it is on the convex hull or if there is an empty quadrilateral that has e as its diagonal and is either not convex or has e as the shorter of its diagonals. Such a quadrilateral is called a *certificate* for e. Edges which do not have a certificate can be eliminated. This can cause other certificates to fail, and one can iterate the certificate-checking until the set of remaining edges stabilizes. The set of remaining edges which are not intersected by any other remaining edge forms a subset of the MWT. It is called the *LMT-skeleton*. The remaining edges which are intersected by another remaining edge are called the *candidate edges*. Several implementations of the LMT heuristic exist. For our computations we follow Beirouti and Snoeyink [2]. Other implementations are described by Cheng, Katoh, and Sugai [7] as well as Hainz, Aichholzer, and Aurenhammer [16].

In this work, we use the LMT-skeleton heuristic to compute MWTs for our gadgets.

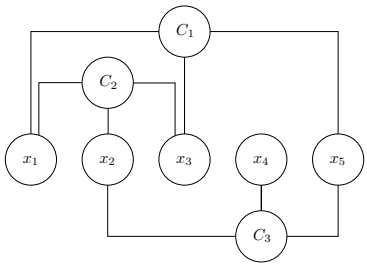

Figure 1: A rectilinear embedding of the associated graph for the Boolean formula $(x_1 \vee \neg x_3 \vee x_5) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee \neg x_5)$.

2.2 PLANAR-1-IN-3-SAT

DEFINITION 1. *Let Φ be a Boolean formula in 3-CNF. The* associated graph *of Φ, $G(\Phi)$, has one vertex v_x for each variable x in Φ and one vertex v_C for each clause C in Φ. There is an edge between a variable-vertex v_x and a clause-vertex v_C if and only if x or $\neg x$ appears in C.*

The Boolean formula Φ is called planar *iff its associated graph $G(\Phi)$ is planar.*

Lichtenstein [22] showed that 3-SAT remains NP-complete if the input is restricted to a planar formula (the PLANAR-3-SAT problem). As Knuth and Raghunathan [21] observed, it follows from Lichtenstein's proof that it suffices to consider formulae whose associated graph can be embedded such that the variables are arranged on a straight line, with three-legged clauses above and below them. The edges between the variables and the clauses are embedded in a rectilinear fashion (see Figure 1).

In our reduction we will use a variant of PLANAR-3-SAT in which we ask for an assignment to the variables such that in each clause exactly one literal is set to true.

DEFINITION 2. *In the* PLANAR-1-IN-3-SAT *problem we are given a collection Φ of clauses containing exactly three literals together with a planar embedding of the associated graph $G(\Phi)$ as described above.*

The problem is to decide whether there exists an assignment of truth values to the variables of Φ such that exactly one literal in each clause is true.

PROPOSITION 1. *PLANAR-1-IN-3-SAT is NP-complete.*

PROOF. We describe a reduction from PLANAR-3-SAT. For that, we describe how to replace a clause $C = (x \vee y \vee z)$ with literals x, y, z by 1-in-3 clauses in such a way that the associated graph remains planar. The clause C can be replaced by

$$(x, \gamma, a) \wedge (y, \gamma, b) \wedge (a, b, q) \wedge (\neg\gamma, \beta, c) \wedge (z, \beta, d) \wedge (c, d, r) \wedge (\beta, e, f) \wedge (\beta, \neg e, \neg f)\,.$$

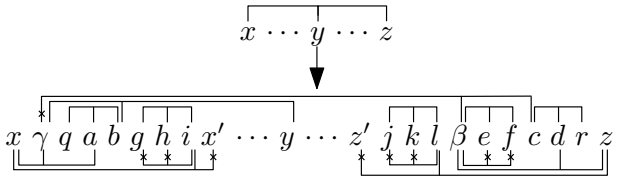

Figure 2: The replacement for a CNF clause with three literals. Negations are indicated by little crosses.

Here, we denote a 1-IN-3 clause by a triple (l_1, l_2, l_3). The clause (l_1, l_2, l_3) is satisfied if exactly one of the literals l_1, l_2, l_3 is true.

LEMMA 1. *Given x, y, and z, the expression* (1) *is satisfiable iff $x \vee y \vee z$ holds.*

We need to add additional variables and clauses to make sure that x and z are reachable from all other clauses. More precisely, we add variables g, h, i, x' and variables j, k, l, z' and clauses $(g, h, i) \wedge (\neg g, \neg h, i) \wedge (x, i, \neg x')$ and $(j, k, l) \wedge (\neg j, \neg k, l) \wedge (z, l, \neg z')$. These clauses ensure $x = x'$ and $z = z'$:

LEMMA 2. *$(g, h, i) \wedge (\neg g, \neg h, i) \wedge (x, i, \neg x')$ is satisfiable iff x and x' have the same value.*

Figure 2 shows the 1-in-3 clauses that replace a clause $C = (x \vee y \vee z)$ with three literals x, y, and z. In a satisfying assignment, the new variables x' and y' have the same truth values as x and y, and they can be used to access the values of x and y from other clauses in the planar embedding. □

3. THE GADGETS

In this section we describe the gadgets which are needed in our reduction. Our gadgets require a large number of points, and we rely on computer assistance to verify them. We implemented the LMT heuristic for minimum weight triangulations as described by Beirouti and Snoeyink [2] in C++ as an Ipelet for Otfried Cheong's extensible drawing editor Ipe[1] [31], which we used as a front-end and user interface. For the calculations we used double-precision floating point arithmetic. We verified the results of our heuristic using an independent implementation of the LMT heuristic by Reinhard Hainz from TU Graz, prepared under the supervision of Oswin Aichholzer. The source code for our implementation and XML files containing the exact coordinates of our gadgets are available on the Internet.[2] The coordinates are given as precise decimal numbers which are multiples of 0.000001. The scale was chosen for convenient representation of the gadgets in Ipe.

The gadgets will be composed in such a way that they simulate a PLANAR-1-IN-3-SAT formula, so the figures shown in this section show details of the overall construction. The optimal triangulations we show were computed only for the local gadgets. The part of the triangulation that is of interest is inside the wires, the outer triangles can be ignored. We will frequently indicate this by shading the unimportant parts.

[1] http://ipe.compgeom.org

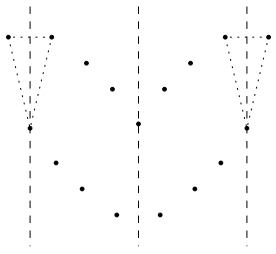

Figure 3: A wire piece. The wire piece can be extended arbitrarily by mirroring the point set along the dashed lines.

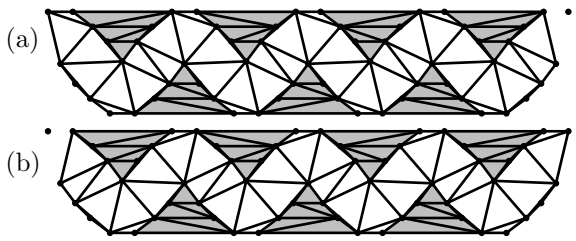

Figure 4: The two states that can be transported by the wire. There are several optimal triangulations of the shaded parts filling up the space around the wire; these triangulations are irrelevant for the state.

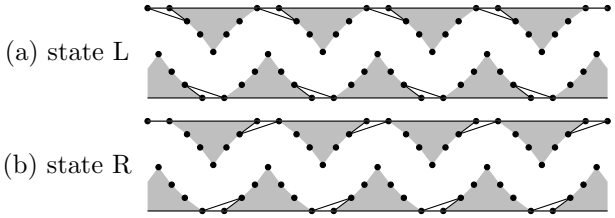

Figure 5: Symbolically, we refer to the two states by L and R, according to the direction (left or right) in which the triangles incident to the boundary of the wire are inclined when looking from the boundary.

3.1 Wires

A *wire* provides a means to transport information from one location in the plane to another. For our construction we use the wire conceived by Jack Snoeyink (see [2]). Its main building block is the wire piece shown in Figure 3.

The wire piece has a vertical axis of symmetry, and it terminates in two isosceles triangles that are indicated in Figure 3. The wires in all our gadgets can be extended arbitrarily along the terminating triangles by inserting an appropriate repetition of wire pieces. This wire has two symmetric optimal triangulations, which are used to encode

[2] http://www.inf.fu-berlin.de/inst/ag-ti/people/rote/Software/MWT/ . This site also contains the full version of this paper as a technical report, with more illustrations.

input	best triangulation	weight in C_0	weight in C_1	relative to (b)
(a) 01 = RL	RRL (or RLL)	3634.078764 ...	3634.290718 ...	0.004333 ... =: ε_2
(b) 00 = RR	RLR	3634.180409 ...	3634.286385 ...	0
(c) 11 = LL	LRL	3634.180409 ...	3634.286385 ...	0
(d) 10 = LR	LLR (or LRR)	3634.290543 ...	3634.290543 ...	0.004158 ... =: ε_1
(b′), (c′)	RRR* or LLL*	3634.188899 ...	3634.294876 ...	0.008491 ... = $\varepsilon_1 + \varepsilon_2$

Table 1: The weights for the four triangulations in Figure 8, for the initial configuration (C_0) and after the perturbation (C_1). The numbers represent the total length of the edges *inside the wires* only. The two triangulations in the last line are marked* because they are not optimal.

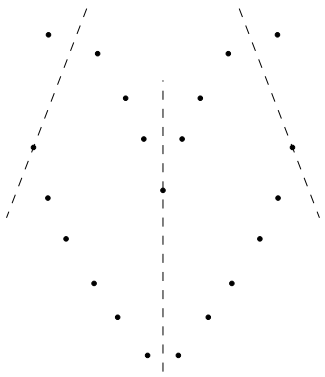

Figure 6: A bent wire piece. The central dashed line shows the axis of symmetry, and the dashed lines at the boundary indicate where other wire pieces will be connected.

logical values and transport them over long distances. We will call these two states L and R (see Figures 4–5). The state L will be used to represent the truth value 1 (true), the state R will correspond to 0 (false). Note that a rotation of a wire by 180° leaves the wire in the same state.

All our gadgets will have the same *terminal triangles* at which they can be connected. The terminal triangles occur in two sizes (for "thick" and "thin" wires) and four orientations (rotated by multiples of 90°). (An exception are the resizing gadgets in Section 3.5, with also occurs in intermediate sizes.)

The wire has some flexibility in its construction. It can be bent to achieve a turn by 90 degrees, By connecting straight wires and right angle turns, we can transport state from one location in the plane to another. By closing a wire into a loop, we get a gadget that has two optimal triangulations of equal weight, thus representing two possible states of a logical variable. All this has been known before [2]. In the next sections we shall discuss how to connect variables to wires and how to construct the clauses.

3.2 Connecting Two Wires

In this section we describe how to connect two wires. As we mentioned above, it is possible to bend a wire piece. When we take this operation to an extreme, we obtain the wire piece shown in Figure 6.

By adding additional points adjacent to the corner of the wire piece, it becomes possible to "break out" through the boundary and to transmit information out of the wire segment. Figure 7 shows how to do this. The small wire leaving

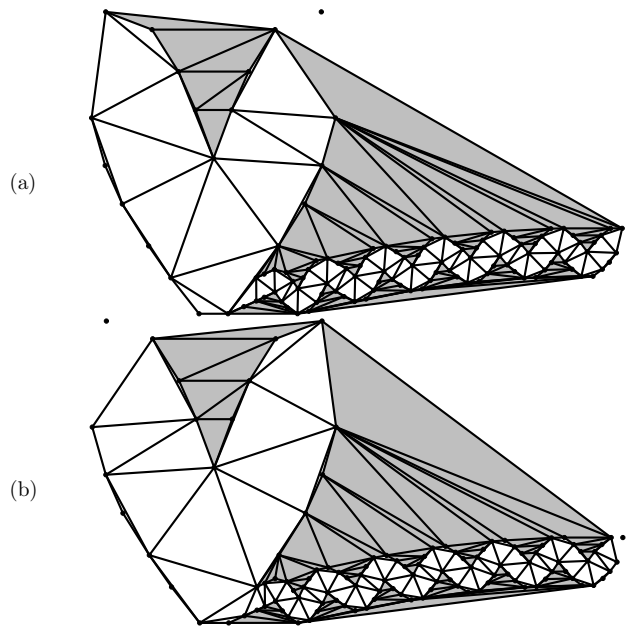

Figure 7: Transporting information out of the wire into a smaller attached wire.

the corner ends in a thinner wire scaled down by a factor of about 1/5. We will refer to wires of the two sizes shown in Figure 7 as *thick wires* and *thin wires*. Thus, we can use the thin wire in order to transport information from one thick wire to another as shown in Figure 8. We will refer to this gadget as the C_0 connection. The wire pieces on the left and right side represent pieces of two thick wires, For each of the four combinations of states of the two thick "input" wires, we can compute the shortest triangulation. Table 1 which shows the lengths of the respective triangulations inside the wire, up to the first ten significant digits. The intermediate thin wire is also in one of the two states L and R, which is indicated by a small letter in the table. Configurations RR and LL are symmetric to each other, and the configuration of the thin wire is forced by the triangulations of the thick wires. In configurations RL and LR, the symmetric triangulation of the thin wire is also possible.

According to Table 1, the most favorable configuration is RL. In order to enforce an equality constraint between the inputs, we want to favor configurations RR and LL over the other two configurations. Thus we penalize the state R for the left input by moving the points a along the line $\overline{0a}$ by a small distance (approximately 0.4558), and symmetrically,

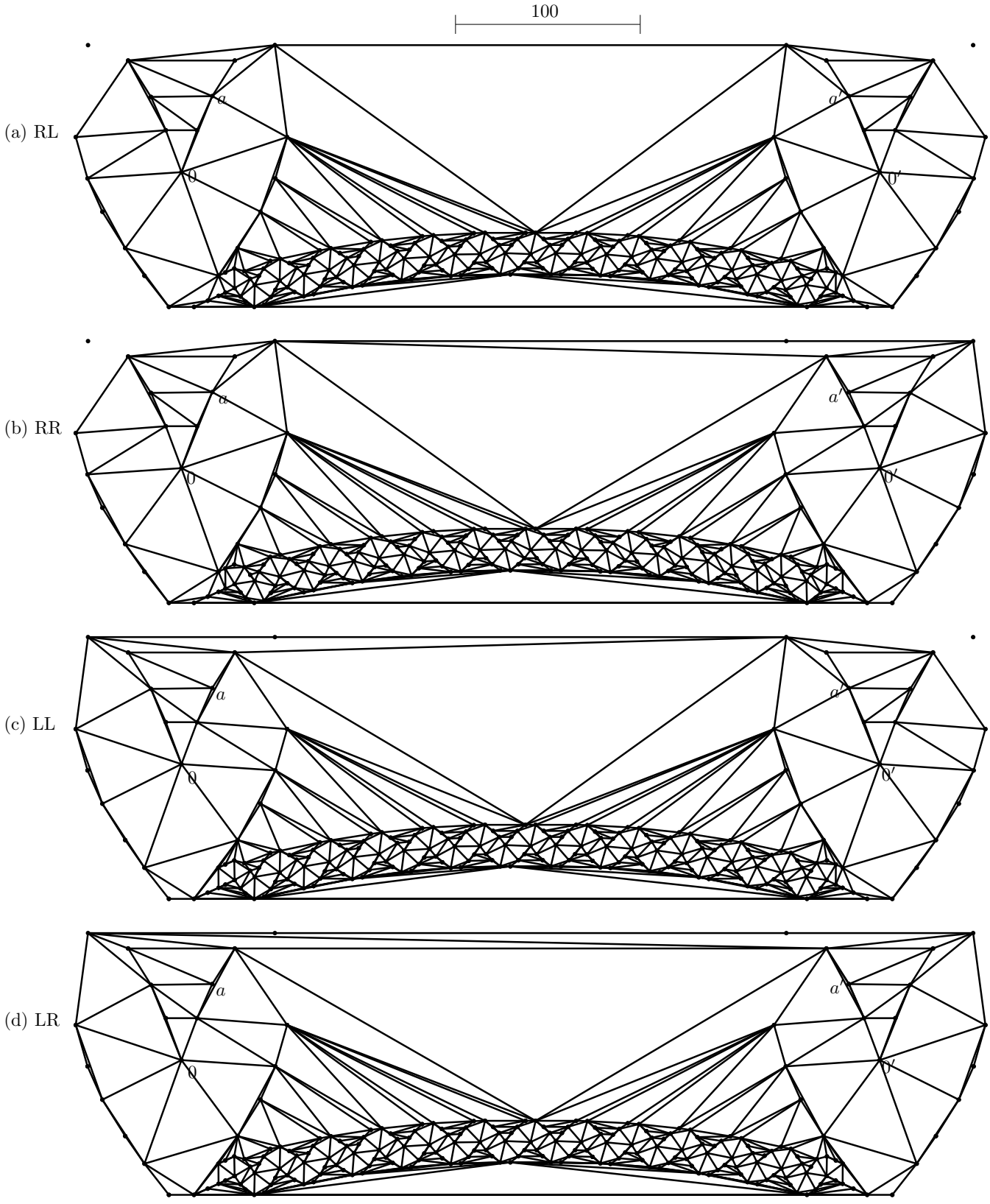

Figure 8: The four ways to connect two wires.

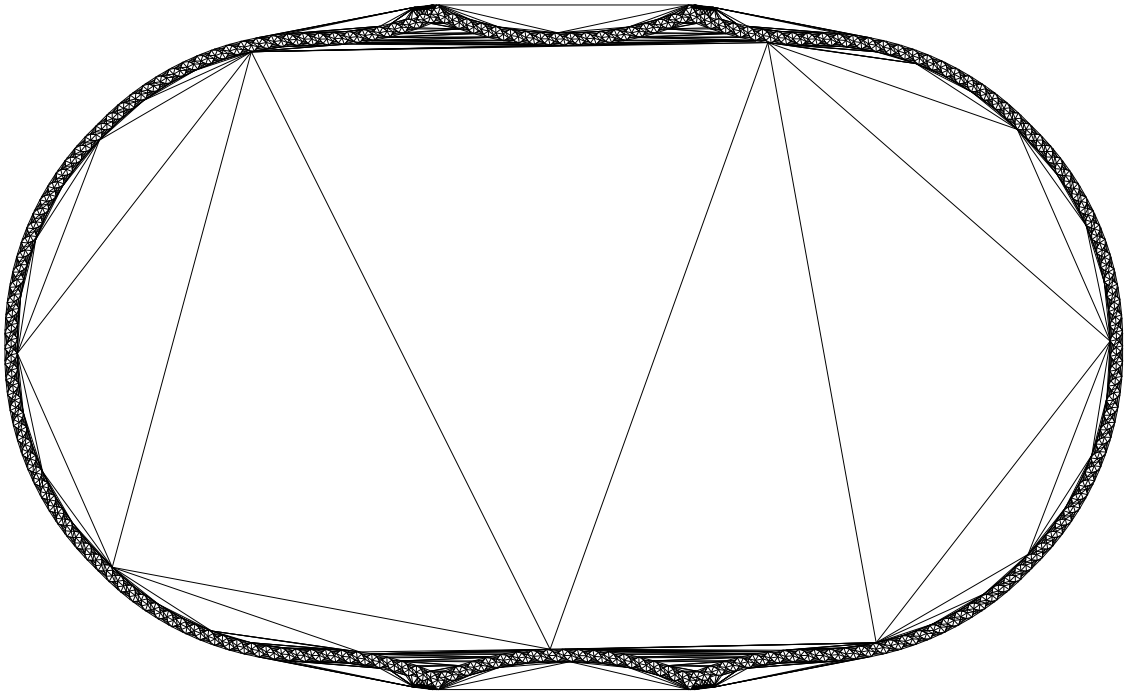

Figure 9: A storage loop in state L (storing 1). This building block has four symmetric "ports" for C_0 or C_1 connections. See Figure 10 for a close-up view.

we penalize the state L for the right input by moving a' along $\overline{0'a'}$. (The precise amount of movement is not important.) This increases the weight of configurations RL, RR, and LL, and configuration RL receives twice as much penalty. The resulting weights are shown in Table 1. Now configurations RR and LL are cheapest. We call this gadget the C_1 connection. By inserting wire pieces and 90° turn into the thin wire, we can construct more complicated forms of this gadget.

3.3 Storage Loops

A *storage loop* is our basic building block for storing a bit. It is composed from right angle gadgets from Section 3.1 and C_1 connections from Section 3.2, see Figures 9–10.

LEMMA 3. *In an optimal triangulation, the thick wire of a storage loop is always uniformly triangulated, either in the L state or in the R state.*

PROOF. This can be checked by a computer calculation. We know that the LMT-skeleton, and thus the wire boundaries must form part of the optimal triangulation.

Let us take a closer look at the LMT-skeleton at some section of a wire (see Figure 11). Every minimum-weight triangulation must either use edge x or edge y. Thus we just have to look at a small number of possibilities (a binary choice for each thick and thin wire occurring in the gadget) and calculate the optimum triangulation for each possibility.

Moreover, if the triangulation uses edge x, two edges x' and x'' that form a triangle with x are also forced to be in the triangulation. Thus, in either case we have an edge that connects the two boundaries of the wire. We therefore get a connected subgraph and can compute the optimum triangulation by dynamic programming. □

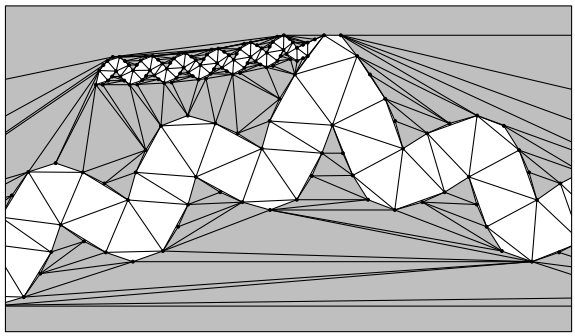

Figure 10: A close-up view of the upper connection in the storage loop of Figure 9. This is the beginning of a thin wire that can connect the storage loop to other loops. (Like most illustrations in this paper, this figure shows the optimal triangulation of the point set which is a section of an intended larger construction, as it was calculated by the computer.)

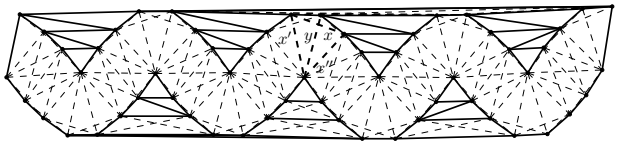

Figure 11: The LMT-skeleton of a part of the storage loop. Exactly one of edges x and y is included in the MWT. If x is included, also x' and x'' will be forced into the MWT, so in any case the LMT-skeleton becomes connected.

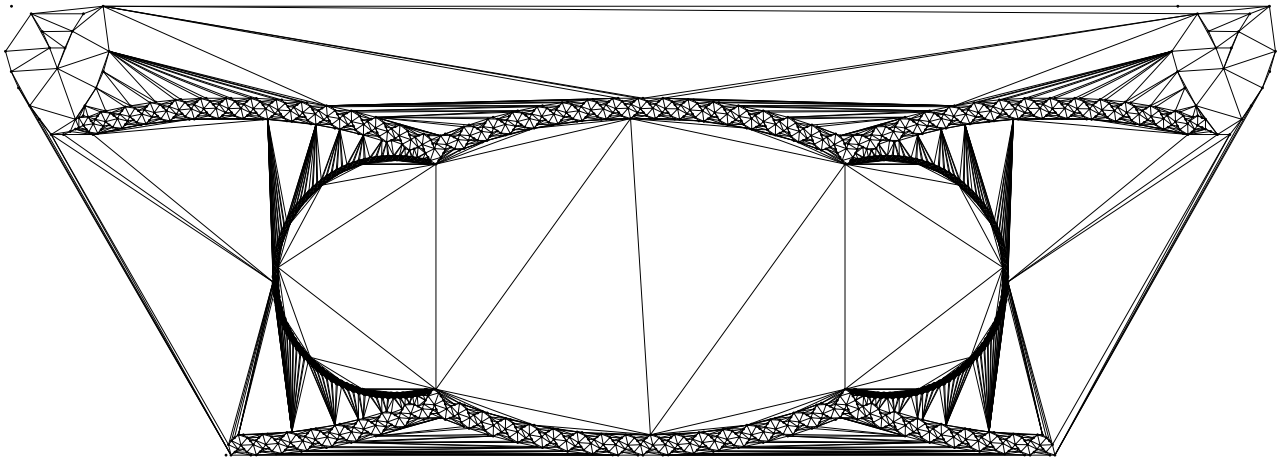

Figure 13: Extracting the state from the thin wire (state 0).

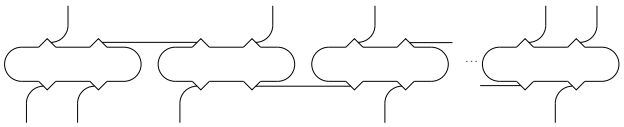

Figure 12: Building a variable from the storage loops.

3.4 Variables

A variable is constructed by linking together an appropriate number of storage loops with thin wires, as shown in Figure 12. In each storage loop — except the first and last one — two C_1 connections are used to connect the storage loops together and two C_1 connections are available to transport the information out of the variables to the clauses. If an exit to the top or the bottom of the variable is not needed, we replace the C_1 connection by the simple C_0 bent wire piece without an attached thin wire.

By Lemma 3, there are two possible ways to triangulate each storage loop, but the C_1 connections ensure that all storage loops are triangulated in the same way, since otherwise the weight of the triangulation increases.

The following lemma is established in the same way as Lemma 3.

LEMMA 4. *In an optimal triangulation, a thin wire connecting two storage loops is always uniformly triangulated, either in the L state or in the R state* (*no matter how the thin wire is composed of straight pieces and* 90° *turns.*) □

From Table 1 we see that the penalty for a mismatched C_1 connection is at least $\varepsilon_1 \approx 0.004158$.

3.5 Negation

Now let us turn to negation gadgets. In configurations LL and RR in Figure 8, the thin wire contains the negation of the value transported in the thick wire. Thus, we need to extract the value from the thin wire and transfer it into another thick wire. We do this by copying the information from the thin wire to a sequence of wires that gradually become thicker and thicker.

First, we extract the state from the thin wire, this is shown in Figure 13. Here, we insert two thin copies of the original C_0 connection into the thin wire that connects two thick wires (which may be part of some storage loop). The two C_0 connections are connected to two C_0 connections on another copy of the thin wire by a "micro-wire" (which appears like a very dense circular arc in the picture). The C_0 connections are arranged in such a way that either we get a combination of configurations RL and LR or a combination of configurations LL and RR. As can be seen in Table 1, the latter combination is cheaper and thus ensures that the two thin wires are in the same configuration. We chose to connect the two wires in this way instead of using the C_1 connections, because this configuration turns out to be more stable.

Having extracted the state from the thin wire, we transport it into a thick wire using a resizing gadget. The resizing loop is almost identical to the storage loop, except that the right or left half is enlarged by a factor of 1.001. The whole point set is still symmetric with respect to the horizontal axis, and there are two optimal states as in Figure 9. The construction has some flexibility, and such a set can be constructed for any rational factor between 1 and 1.001.

By composing an appropriate number of these devices, it is possible to scale up the wire and its state to any desired degree. This is outlined in Figure 14. As with the original storage loop, there are exactly two possible ways to triangulate the resizing loops, and the thin wires ensure that the triangulations are consistent (Lemmas 3 and 4).

LEMMA 5. *Let K be the cost of the optimal triangulation within all wires if the two upper blocks in Figure 14 are in the same state and the lowest block is in the opposite state. If these three blocks are in any other combination of states, then the cost is at least $K + 10^{-4}$.* □

The reason for resizing the wire in this seemingly complicated way is that each individual loop is in a stable state, and thus one can analyze the components of the gadget (the loops and the C_1 connections) in isolation. Our initial attempts to have a single wire that gradually increases its width failed because we could not limit the interaction between larger and smaller pieces in a long wire whose width increases gradually.

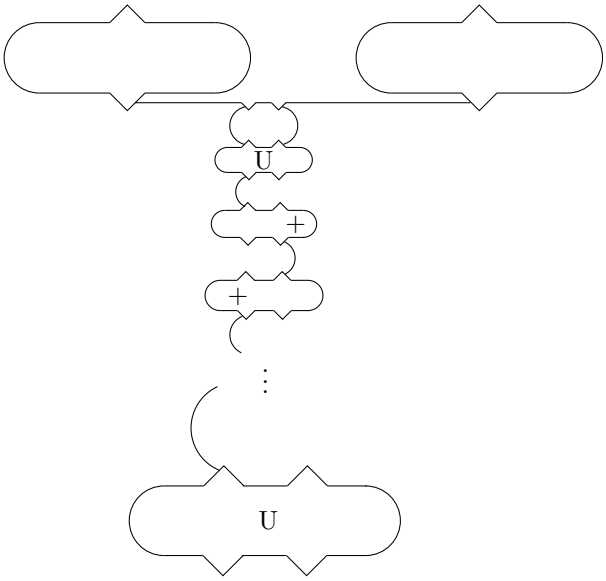

Figure 14: Schematic outline of a track of the negation gadget. The detailed picture of the upper middle part is shown in Figure 13. For clarity, the size increase between successive stages has been exaggerated. The enlarged half of the resizing loop (+) is alternately on the left and on the right hand side of the gadget and connects to the next resizing loop in the chain. The first and the last loop have uniform size (U).

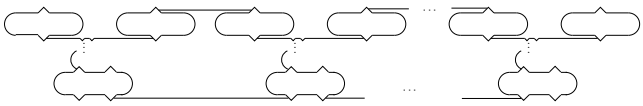

Figure 15: Outline of the complete negation gadget, connecting many parallel tracks of Figure 14 in a row.

Now, to boost the penalty for a triangulation that does not enforce the negation, we connect many tracks in parallel as shown in Figure 15. In this gadget, the inputs and outputs of the negation blocks are connected by normal C_1 connections. Thus, all inputs and outputs must be the same, or we will incur the penalty for a mismatched connection. If one of the negation gadgets is not triangulated correctly, we will either incur the penalty for an inconsistency in the outputs or all of the negation gadgets will be incorrect. Then, each of the negation gadgets will be penalized. If we connect sufficiently many negation gadgets, this total penalty will always be larger than the penalty ε_1 for a mismatched connection.

The negation gadget becomes very large and needs several million points, but still the amount of space needed is constant. The resizing loop must be repeated approximately 2000 times to achieve the desired enlargement factor $1.001^{2000} > 5$. The coordinates are multiplied by 1.001 in each step, producing three new decimal digits at the end. One finally arrives at coordinates which are multiples of $\varepsilon_0 := 10^{-6000}$.

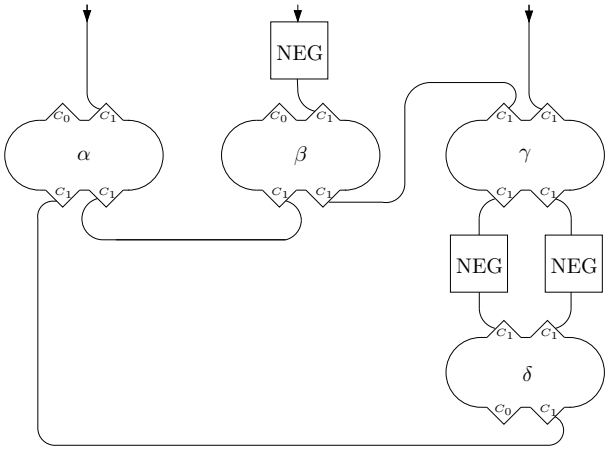

Figure 16: Outline of the clause gadget. The inputs arrive from above.

3.6 Clauses

Using the connection gadgets, we build a clause gadget by connecting four storage loops $\alpha, \beta, \gamma, \delta$ as shown in Figure 16. The connections are annotated according to their type. Let $\varepsilon_1 \approx 0.004158$ and $\varepsilon_2 \approx 0.004333$ be the the two penalties we incur when a C_1 connection is not triangulated correctly. More precisely, we incur a penalty of ε_1 when the "left" wire of the C_1 connection is in state 1 (or L) and the "right" wire is in state 0 (or R), and so on. Note that we can control which wire is the "left" and the "right" input by bending the wire appropriately. Thus, for example, in the C_1 connection between α and β, the "left" wire is β.

LEMMA 6. *The weight of the shortest triangulation of the clause gadget achieves its minimum value W when exactly one of the input wires carries the state* 1. *In any other triangulation, the weight is at least $W + \varepsilon_2 - \varepsilon_1 \approx W + 0.00017$.*

PROOF. We claim than, among the 16 ways to triangulate the four storage loops of the clause gadget, the minimum possible penalty is ε_1. We have connected γ and δ by two parallel negation gadgets. As discussed above, we can ensure that the penalty for a mis-triangulated negation gadget is at least ε_1. Thus, if δ and γ are not opposite, the penalty is at least $2\varepsilon_1 > \varepsilon_1$. Table 2 lists all possibilities with $\delta \neq \gamma$ and the according penalties. The cheapest configurations are 0001, 0110, and 1101, with penalty ε_1, as desired. □

α	β	γ	δ	penalty
0	0	0	1	ε_1
0	0	1	0	ε_2
0	1	0	1	$3\varepsilon_1$
0	1	1	0	ε_1
1	0	0	1	ε_2
1	0	1	0	$3\varepsilon_2$
1	1	0	1	ε_1
1	1	1	0	ε_2

Table 2: The different ways to triangulate the clause gadget.

4. THE REDUCTION

THEOREM 1. *Minimum weight triangulation is strongly NP-hard.*

PROOF. A rectilinear embedding of the given PLANAR-1-IN-3-SAT formula can be constructed on a grid of size $O(n) \times O(n)$. The reduction procedure then simply replaces the edges, variables and clauses of PLANAR-1-IN-3-SAT formula by the appropriate gadgets.

We have constructed gadgets of various (constant) sizes, and for connecting them, we may wish to have wires of arbitrary lengths available. We solve this by using two different wire pieces for the thin wire. We know that all gadgets are embedded on a grid of size $\varepsilon_0 = 10^{-6000}$. So we use a "native" thin wire piece of length $L_1 \leq 30$ from Figure 7 and an extended version of length $L_1 + \varepsilon_0$. These two storage loops can be combined to build wires any length larger than 900×10^{6000}. This length is also more than enough to ensure that edges are long enough to accommodate two 90° turns at their endpoints. Thus, we can realize any rectilinear layout with 90° turn and straight wire pieces, after blowing it up sufficiently.

This procedure yields a point set S. By construction, the boundaries of all wires belong to the minimum-weight triangulation of S. One can argue that the properties that we have proved for single gadgets remain true when we put them together to create the set S: the gadgets which are far away do not interfere with each other since the additional edges would fail the "diamond test" of Das and Joseph [9].

The faces outside the wires are simple polygons and can be triangulated using dynamic programming. For each gadget, we know the desired "ideal triangulation" and can calculate its weight. Adding up these weights and the weights of the faces outside the wires yields a target weight W. By construction, the input instance is 1-IN-3 satisfiable iff the minimum weight of a triangulation of S is W. Otherwise, the weight of the shortest triangulation is at least $W + 0.00017$. (The smallest penalty for not satisfying a clause or otherwise violating one of the consistence conditions is the difference between ε_2 and ε_1 in Table 2, see Lemma 6.)

The set S has $O(n^2)$ points (being a subset of an $O(n) \times O(n)$ grid), and hence the triangulation has $O(n^2)$ edges. By calculating all edge lengths with an absolute error of $O(1/n^2)$, the reduction algorithm can thus calculate, in polynomial time, a threshold $\hat{W}$ such that the input formula is satisfiable iff there is a triangulation of length at most $\hat{W}$. □

5. CONCLUSION

Several interesting open problems remain. First of all, it is not known whether the MWT problem is in NP, since it is not known how to compare sums of euclidean lengths in polynomial time. To define a variant of MWT which is in NP, one can take the weight of an edge e as the rounded value $\lceil \|e\|_2 \rceil$. With appropriate scaling, our proof also establishes NP-completeness for this variant.

Our result shows that it is NP-hard to approximate the MWT with a relative approximation error which is better than $O(1/n^2)$. This does not rule out the existence of a PTAS. Until very recently, attempts to extend techniques from geometric approximation algorithms to the MWT problem have only led to constant factor approximations (see [3] for a survey). Remy and Steger [30] showed that it is possible to compute a $(1+\varepsilon)$-approximation of the MWT in time $n^{O(\log^8 n)}$, providing strong evidence that a PTAS might exist.

Our current proof relies on (heavy) computations involving floating-point numbers. However, after the initial calculation of edge lengths, the computations involve only additions and comparisons of a few hundred positive values, and thus the numerical error is well under control. Still, the construction of the gadgets has required some delicate fine-tuning. We are working towards more "localized" arguments that would not need to consider a storage loop as a whole, as well as on independent and more reliable verifications of our constructions, using interval arithmetic.

By adding additional points, we hope to modify the gadgets in such a way that it is sufficient to use the β-skeleton for proving that the walls of the wires and gadgets belong to the MWT. This would provide a more directly accessible (even visual) correctness proof of the gadgets than the LMT-skeleton, which requires a computer program.

6. REFERENCES

[1] E. Anagnostou and D. Corneil. Polynomial time instances of the minimum weight triangulation problem. *Computational Geometry: Theory and Applications* **3** (1993), 247–258.

[2] R. Beirouti and J. Snoeyink. Implementations of the LMT heuristic for minimum weight triangulations. *Proc. 14th Ann. Symp. Computat. Geometry*, Minneapolis, pp. 96–105, 1998.

[3] M. Bern and D. Eppstein. Approximation algorithms for geometric problems in *Approximation Algorithms for NP-Hard Problems*, ed. D. S. Hochbaum, PWS Publishing Company, Boston, Mass., 1997, pp. 296–345.

[4] P. Belleville, M. Keil, M. McAllister, and J. Snoeyink. On computing edges that are in all minimum weight triangulations. *Proceedings of the 12th Annual Symposium on Computational Geometry*, Philadelphia, pp. V7–V8, 1996.

[5] P. Bose, L. Devroye, and W. Evans. Diamonds are not a minimum weight triangulation's best friends. *Proceedings of the 8th Canadian Conference on Computational Geometry*, Ottawa, pp. 68–73, 1996.

[6] S. W. Cheng, M. J. Golin, and J. C. F. Tsang. Expected case analysis of β-skeletons with applications to the construction of minimum weight triangulations. *Proceedings of the 8th Canadian Conference on Computational Geometry*, Ottawa, pp. 279–284, 1996.

[7] S. W. Cheng, N. Katoh, and M. Sugai. A study of the LMT-skeleton. *Proceedings of the 7th Annual International Symposium on Algorithms and Computation* , Osaka, Japan. Springer Lecture Notes in Computer Science, volume 1178, pp. 256–265, 1996.

[8] S. W. Cheng and Y. F. Xu. Approaching the largest β-skeleton within a minimum weight triangulation. *Proceedings of the 12th Annual Symposium on Computational Geometry*, Philadelphia, Pennsylvania, pp. 196–203, 1996.

[9] G. Das and D. Joseph. Which triangulations approximate the complete graph. *Proceedings of the International Symposium on Optimal Algorithms*, Varna, Bulgaria. Springer Lecture Notes in Computer Science, volume 401, pp. 168–192, 1989.

[10] M. T. Dickerson, J. M. Keil, and M. H. Montague. A large subgraph of the minimum weight triangulation. *Discrete and Computational Geometry* **18** (1997), 289–304.

[11] R. L. Drysdale, S. A. McElfresh, and J. Snoeyink. On exclusion regions for optimal triangulations. *Discrete Applied Mathematics* **109** (2001), 49–65.

[12] R. L. Drysdale, G. Rote, and O. Aichholzer. A simple linear time greedy triangulation algorithm for uniformly distributed points. *IIG-Report-Series 408*, Technische Universität Graz, 1995.

[13] R. D. Duppe and H. J. Gottschalk. Automatische Interpolation von Isolinien bei willkürlichen Stützpunkten. *Allgemeine Vermessungsnachrichten* **77** (1970), 423–426.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, New York, New York, 1979.

[15] P. D. Gilbert. New results in planar triangulations. Report R-850, Coordinated Science Laboratory, University of Illinois, Urbana, Illinois, 1979.

[16] R. Hainz, O. Aichholzer, and F. Aurenhammer. New results on minimum weight triangulations and the LMT-skeleton. *Proceedings of the 13th European Workshop on Computational Geometry*, Würzburg, Germany, pp. 4–6, 1997.

[17] M. Hoffmann and Y. Okamoto. The minimum weight triangulation problem with few interior points. *Proceedings of the International Workshop on Parameterized and Exact Computation*, Bergen, Norway, Springer Lecture Notes in Computer Science, volume 3162, pp. 200–212, 2004.

[18] D. Johnson. The NP-completeness column. *ACM Trans. Algorithms* **1** (2005), 160–176.

[19] J. M. Keil. Computing a subgraph of the minimum weight triangulation. *Computational Geometry: Theory and Applications* **4** (1994), 13–26.

[20] G. T. Klincsek. Minimal triangulations of polygonal domains. *Annals of Discrete Mathematics*, volume 9, 1980, pp. 121–123.

[21] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics* **5** (1992), 422–427.

[22] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Computing* **11** (1982), 329–343.

[23] D. G. Kirkpatrick. A note on Delaunay and optimal triangulations. *Information Processing Letters* **10** (1980), 127–128.

[24] Y. Kyoda, K. Imai, F. Takeuchi, and A. Tajima. A branch-and-cut approach for minimum weight triangulation. *Proceedings of the 8th International Symposium on Algorithms and Computation*, Singapore. Springer Lecture Notes in Computer Science, volume 1350, pp. 384–393, 1997.

[25] C. Levcopoulos. An $\Omega(\sqrt{n})$ lower bound for the nonoptimality of the greedy triangulation. *Information Processing Letters* **25** (1987), 247–251.

[26] C. Levcopoulos and D. Krznaric. Quasi-greedy triangulations approximating the minimum weight triangulation. *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, Atlanta, Georgia, 1996, pp. 392–401.

[27] E. L. Lloyd. On triangulations of a set of points in the plane. *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science*, Providence, Rhode Island, pp. 228–240, 1977.

[28] G. K. Manacher and A. L. Zobrist. Neither the greedy nor the the Delaunay triangulation approximates the optimum. *Information Processing Letters* **9** (1979), 31–34.

[29] D. A. Plaisted and J. Hong. A heuristic triangulation algorithm. *Journal of Algorithms* **8** (1987), 405–437.

[30] J. Remy and A Steger. A quasi-polynomial time approximation scheme for minimum weight triangulation. *Proceedings of the 38th ACM Symposium on Theory of Computing*, Seattle, Washington, 2006.

[31] O. Schwarzkopf. The extensible drawing editor Ipe. *Proceedings of the 11th Annual Symposium on Computational Geometry*, Vancouver, British Columbia, pp. 410–411, 1995.

[32] M. I. Shamos and D. Hoey. Closest point problems. *Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science*, Berkeley, California, pp. 151–162, 1975.

[33] B. T. Yang, Y. F. Xu, and Z. Y. You. A chain decomposition algorithm for the proof of a property on minimum weight triangulations. *Proceedings of the 5th Annual International Symposium on Algorithms and Computation*, Beijing, China. Springer Lecture Notes in Computer Science, volume 834, pp. 423–427, 1994.