



# Exact Algorithms for Minimum Dilation Triangulation

Sándor P. Fekete  

Department of Computer Science, TU Braunschweig, Germany

Phillip Keldenich  

Department of Computer Science, TU Braunschweig, Germany

Michael Perk  

Department of Computer Science, TU Braunschweig, Germany

---

## Abstract

We provide a spectrum of new theoretical insights and practical results for finding a Minimum Dilation Triangulation (MDT), a natural geometric optimization problem of considerable previous attention: Given a set  $P$  of  $n$  points in the plane, find a triangulation  $T$ , such that a shortest Euclidean path in  $T$  between any pair of points increases by the smallest possible factor compared to their straight-line distance. No polynomial-time algorithm is known for the problem; moreover, evaluating the objective function involves computing the sum of (possibly many) square roots. On the other hand, the problem is not known to be NP-hard.

(1) We provide practically robust methods and implementations for computing an MDT for benchmark sets with up to 30,000 points in reasonable time on commodity hardware, based on new geometric insights into the structure of optimal edge sets. Previous methods only achieved results for up to 200 points, so we extend the range of optimally solvable instances by a factor of 150.

(2) We develop scalable techniques for accurately evaluating many shortest-path queries that arise as large-scale sums of square roots, allowing us to certify exact optimal solutions, with previous work relying on (possibly inaccurate) floating-point computations.

(3) We resolve an open problem by establishing a lower bound of 1.44116 on the dilation of the regular 84-gon (and thus for arbitrary point sets), improving the previous worst-case lower bound of 1.4308 and greatly reducing the remaining gap to the upper bound of 1.4482 from the literature. In the process, we provide optimal solutions for regular  $n$ -gons up to  $n = 100$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Theory of computation  $\rightarrow$  Discrete optimization; General and reference  $\rightarrow$  Experimentation; Mathematics of computing  $\rightarrow$  Interval arithmetic; Mathematics of computing  $\rightarrow$  Arbitrary-precision arithmetic

**Keywords and phrases** dilation, minimum dilation triangulation, exact algorithms, algorithm engineering, experimental evaluation

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2025.48

**Related Version** *Full Version*: <https://arxiv.org/abs/2502.18189>

**Supplementary Material** *Software (Source Code)*: <https://doi.org/10.5281/zenodo.14266122>

*Dataset (Experiment Data)*: <https://doi.org/10.5281/zenodo.14266122>

**Funding** The work presented in this paper was largely funded by the DFG grant “Computational Geometry: Solving Hard Optimization Problems” (CG:SHOP), grant FE 407/21-1.

## 1 Introduction

Triangulating a set of points is one of the classical problems in computational geometry. On the practical side, it has natural applications in wireless sensor networks [47, 49], mesh generation [4], computer vision [45], geographic information systems [46] and many other areas [15]. On the theoretical side, finding a triangulation that is optimal with respect to



© Sándor P. Fekete, Phillip Keldenich, and Michael Perk;  
licensed under Creative Commons License CC-BY 4.0

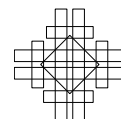
41st International Symposium on Computational Geometry (SoCG 2025).

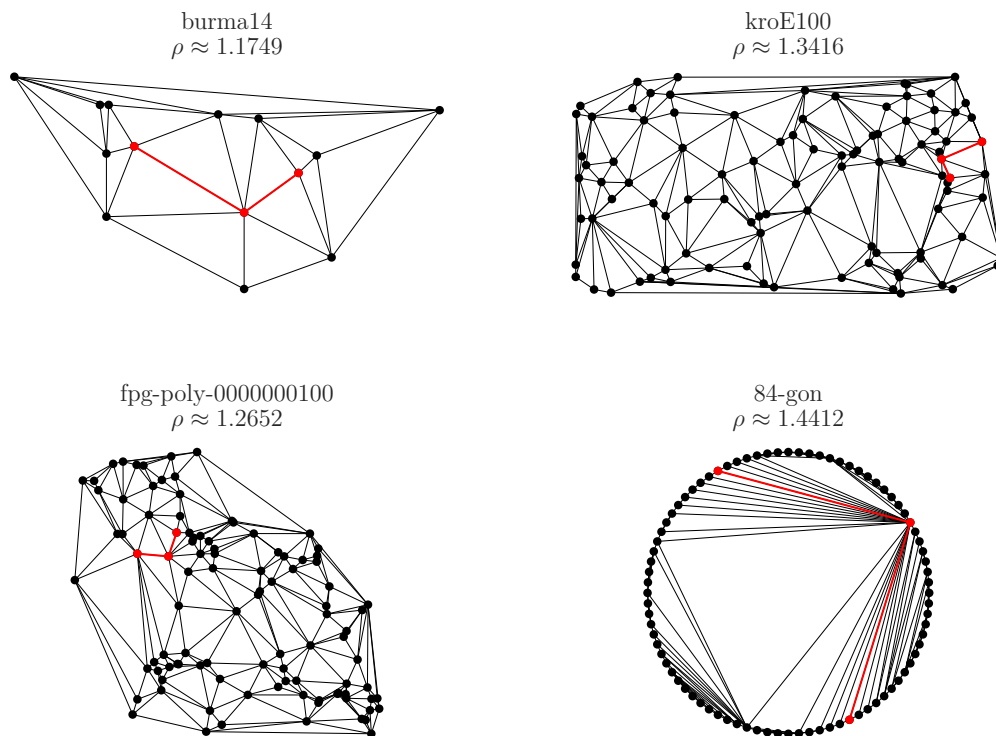
Editors: Oswin Aichholzer and Haitao Wang; Article No. 48; pp. 48:1–48:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** MDT solutions for four instances. The red edges indicate a dilation-defining path.

some objective function has also received considerable attention: The Delaunay triangulation maximizes the minimum angle and minimizes the maximum circumcircle of all triangles. Minimizing the maximum edge length is possible in quadratic time [21]. On the other hand, maximizing the minimum edge length is NP-complete [27]. Famously, Mulzer and Rote [36] showed that computing the Minimum Weight Triangulation (MWT) is NP-hard.

In this paper, we provide new results and insights for another natural objective that considers triangulations as sparse structures with relative low cost for ensuing detours: The *dilation* of a triangulation  $T$  of a point set  $P$  is the worst-case ratio (among all  $s, t \in P$ ) between the shortest  $s$ - $t$ -path in  $T$  and the Euclidean distance between  $s$  and  $t$ . The Minimum Dilation Triangulation (MDT) problem asks for a triangulation that minimizes the dilation  $\rho$ , see Figure 1 for examples. This problem is closely related to the concept of a Euclidean  $t$ -spanner: a subgraph with dilation (also called *spanning ratio* or *stretch factor* [37]) at most  $t$ . Spanners have application in areas as robotics, network design [1, 26], sensor networks [10, 25] and design of parallel machines [3] and have been studied extensively [11]. Computing the MDT amounts to computing a *plane* spanner with smallest spanning ratio, as every plane spanner can be extended into a triangulation. For many types of triangulations, such as the Delaunay triangulation or the MWT, both lower and upper bounds on the worst-case dilation are known. Moreover, a constant upper bound on the worst-case dilation of a triangulation also implies a constant-factor approximation of the MDT. Lower and upper bounds on the worst-case dilation of the MDT have been studied, both for general point sets and for special point sets such as regular polygons or points on a circle; see Section 1.2 for further details.

Despite this importance and attention, actually computing a Minimum Dilation Triangulation is a challenging problem. Its computational complexity is still unresolved, signaling that there may not be a simple and elegant algorithmic solution that scales well. Moreover, actually

computing accurate dilations involves computing shortest paths under Euclidean distances between  $\Theta(n^2)$  pairs of points requires dealing with another famous challenge [39, 18, 6, 23], as it corresponds to evaluating and comparing numerous sums of many square roots.

## 1.1 Our contributions

We provide various new contributions, both to theory and practice.

1. We present practically robust methods and implementations for computing an optimal MDT for benchmark sets with up to 30,000 points in reasonable time on commodity hardware, based on new geometric insights into the structure of optimal edge sets. Previous methods only achieved results for up to 200 points (involving one computational routine of complexity  $\Theta(n^4)$  instead of our improved complexity of  $O(n^2 \log n)$ ), so we extend the range of practically solvable instances by a factor of 150.
2. We develop scalable techniques for accurately evaluating many shortest-path queries that arise as large-scale sums of square roots, allowing us to certify exact optimal solutions. This differs from previous work, which relied on floating-point computations, without regard for errors resulting from numerical issues.
3. We resolve an open problem from [20] by establishing a lower bound of 1.44116 on the dilation of the regular 84-gon (and thus for arbitrary point sets). This improves the previous worst-case lower bound of 1.4308 from the regular 23-gon and greatly reduces the remaining gap to the upper bound of 1.4482 from [43]. In the process, we provide optimal solutions for regular  $n$ -gons up to  $n = 100$ .

## 1.2 Related work

The complexity of finding the MDT is unknown [24]. [28] prove that finding the minimum dilation graph with a limited number of edges is NP-hard. [12] show that finding a spanning tree of given dilation is also NP-hard. Kozma [33] proves NP-hardness for minimizing the expected distance between random points in a triangulation, with edge weights instead of Euclidean distances. For surveys, see Eppstein [24] until 2000 and [38] for more recent results.

On the practical side, the authors of [44] study the problem of finding sparse low dilation graphs for large point sets in the plane. The authors of [9] present an approximation algorithm for the related problem of improving a given graph with a budget of  $k$  edges such that the dilation is minimized. Regarding the MDT, all practical approaches in the literature are based on fixed-precision arithmetic. Klein [31] used an enumeration algorithm to find an optimal MDT for up to 10 points. The authors of [19] present heuristics for the MDT and evaluate their performance on instances with up to 200 points. Instances with up to 70 points were solved by the authors of [8] using integer linear programming techniques. In their approach, the edge elimination strategy from Knauer and Mulzer [32] was used to eliminate edges from the complete graph. Recently, Sattari and Izadi [41] presented an exact algorithm based on branch and bound that was evaluated on instances with up to 200 points.

The MDT is closely related to finding a plane  $t$ -spanner; see [34] for an overview. Chew [13] first proved an upper bound of  $\sqrt{10}$  on plane  $t$ -spanners in the  $L_1$ -metric, which he later improved [14] to 2 for the triangular-distance Delaunay graph in the plane. [5] proved that any convex point set admits a plane 1.88-spanner. For a centrally symmetric convex point set containing  $n$  points, Sattari and Izadi [42] give an upper bound of  $n^{1/2} \sin(\pi/n)$ . The best known upper bound on the dilation of arbitrary point sets is by Xia [48], who established an upper bound of 1.998 for the dilation of the Delaunay triangulation.

Mulzer [35] studied the MDT for the set of vertices of a regular  $n$ -gon and proved an upper bound of 1.48586. Amarnadh and Mitra [2] improved this bound to 1.48454 for any point set that lies on the boundary of a circle. Sattari and Izadi [43] again improved the bound to 1.4482. Dumitrescu and Ghosh [20] show that any triangulation of a regular 23-gon has dilation at least 1.4308, improving upon the bound of 1.4161 by Mulzer [35] and answering a question posed by Bose and Smit [7] as well as Kanj [30]. Dumitrescu and Ghosh [20] also computed dilations of regular 22-gon and regular 24-gon.

## 2 Preliminaries

Let  $P \subset \mathbb{R}^2$  be a set of points in the plane. We denote the Euclidean distance between two points  $u, v \in P$  by  $d(u, v)$ . For a connected geometric graph  $G = (P, E)$  with  $E \subseteq \binom{P}{2}$ , we denote the Euclidean shortest path between two points  $u, v \in P$  by  $\pi_G(u, v)$  and its length by  $|\pi_G(u, v)|$ , omitting  $G$  if it is clear from context. The dilation  $\rho_G(u, v)$  between two points  $u, v$  in  $G$  is the ratio  $\rho_G(u, v) := \frac{|\pi_G(u, v)|}{d(u, v)}$  between the shortest path length and the Euclidean distance. The dilation  $\rho(G)$  of the graph  $G$  is defined as the maximum dilation between any two points in  $P$ , i.e.,  $\rho(G) := \max\{\rho_G(u, v) \mid u, v \in P, u \neq v\}$ .

In the remainder of this work, the graph  $G$  we consider is a triangulation, i.e., a maximal crossing-free graph on  $P$ . Two edges  $e_1 = (p_1, q_1), e_2 = (p_2, q_2)$  are said to *cross* or *intersect* iff the line segments they induce intersect in their interior. Given a point set  $P$ , the Minimum Dilation Triangulation problem (MDT) asks to find a triangulation  $T$  of  $P$  minimizing  $\rho(T)$ .

## 3 Candidate edge enumeration

Here we describe a novel and practically efficient scheme for enumerating a set of edges that induces a supergraph of the MDT. We start with the underlying theoretical ideas for this supergraph, followed by an algorithm for computing a supergraph of any triangulation with dilation *strictly less* than a given bound  $\rho$ , which we exploit to enumerate a supergraph of the MDT with a (usually) small number of edges. This is further adapted to reductions of the dilation bound  $\rho$ . We also discuss the computation of lower bounds on the dilation of the MDT. Some additional implementation details can be found in the full version.

### 3.1 Theoretical background

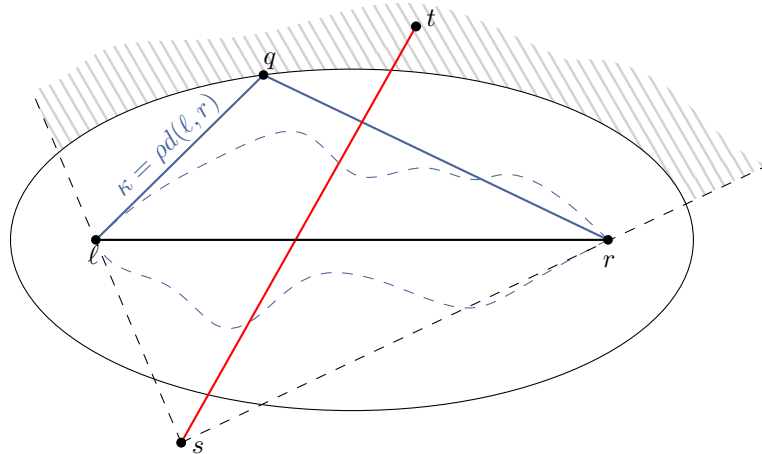
Our supergraph is based on the well-known *ellipse property* (used in [8, 28, 32]) that all edges of a triangulation  $T$  with dilation below  $\rho$  must satisfy.

► **Definition 1.** A pair of points  $s, t$  has the *ellipse property with respect to a point set  $P$  and a dilation bound  $\rho$*  if, for any pair of points  $\ell, r \in P \setminus \{s, t\}$  such that  $\ell r$  and  $st$  intersect,  $\min\{d(\ell, s) + d(s, r), d(\ell, t) + d(t, r)\} < \rho d(\ell, r)$ .

Recall that the set of points that have the same sum of distances  $\kappa$  to two points  $\ell, r$  with  $\kappa \geq d(\ell, r)$  is an *ellipse* with  $\ell, r$  as its focal points (or *foci*); thus, all paths between  $\ell$  and  $r$  with length less than  $\kappa = \rho \cdot d(\ell, r)$  must lie strictly inside this ellipse.

If a pair of points  $s, t$  does not have the ellipse property, then there is a pair of points  $\ell, r$  such that  $st$  cuts through all paths between  $\ell$  and  $r$  that could have dilation less than  $\rho$ ; see Figure 2. We call such a pair of points  $\ell, r$  an *exclusion certificate* for  $s, t$ .

► **Observation 2.** Let  $\rho \geq 1$  be some dilation bound and let  $T$  be a triangulation containing the edge  $st$  for points  $s, t$  that do not have the ellipse property w.r.t.  $P$  and  $\rho$ . Then,  $\rho(T) \geq \rho$ .



■ **Figure 2** Any path connecting  $\ell$  and  $r$  with length below  $\kappa = \rho d(\ell, r)$  must lie within the ellipse (dashed black lines). The edge  $st$  does not have the *ellipse property* as neither  $s$  nor  $t$  lie inside the ellipse; therefore, inserting  $st$  makes connecting  $\ell$  and  $r$  by a sufficiently short path impossible.

### 3.2 High-level description

Preliminary experiments showed that a brute-force check of each of the  $\Theta(n^4)$  pairs of potential edges is impractical for large  $n$ , dominating the overall runtime time even for early versions of our solution approach.

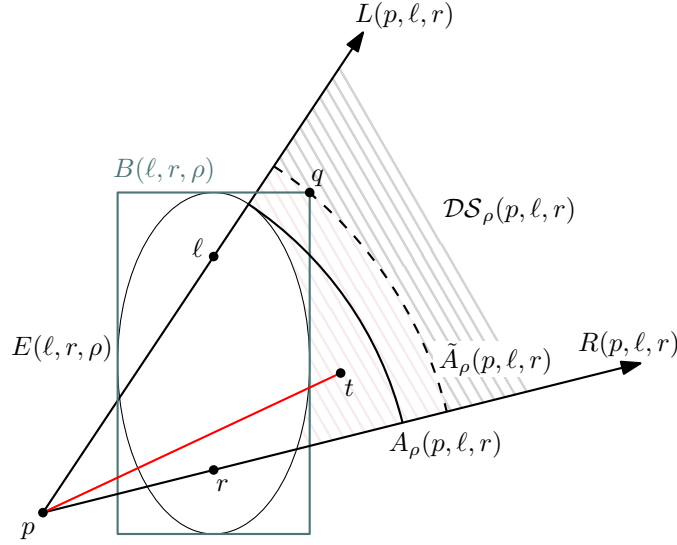
We therefore developed a more efficient scheme to enumerate a superset of the edges that satisfy the ellipse property. This scheme performs a *filtered incremental nearest-neighbor search* from each point  $p \in P$  to identify all candidate edges of the form  $pt$ , i.e., looking for all possible *neighbors*  $t$  of  $p$ . This search is efficiently implemented on a quadtree containing all points. While enumerating candidates, we construct so-called *dead sectors*, i.e., regions of the plane that cannot contain possible neighbors of  $p$ . We exclude all points that lie in dead sectors; we also use dead sectors to prune entire nodes of the quadtree and to terminate the search early if it has become clear that all remaining points must be in a dead sector. This usually avoids considering most points as potential neighbors of  $p$  individually. The algorithm has a runtime of  $O(nk \log n)$ , where  $k$  is the average number of points and quadtree vertices considered individually from each point. At worst, this can be  $O(n^2 \log n)$ ; in practice,  $k$  is often much lower than  $n$ . A related, slightly less complex enumeration algorithm applied to minimum-weight triangulations by Haas [29] scales to  $10^8$  points. In the following, we describe the components of the enumeration scheme in more detail.

### 3.3 Dead sectors

We begin by giving a definition of the dead sectors we use.

► **Definition 3.** Given a dilation  $\rho$ , a source point  $p$  and two points  $\ell, r$ , the dead sector  $\mathcal{DS}_\rho(p, \ell, r) \subset \mathbb{R}^2$  is the region of all points  $t$  such that  $pt$  intersects  $\ell r$  and neither  $p$  nor  $t$  lie in the ellipse with foci  $\ell, r$  and focal distance sum  $\kappa = \rho d(\ell, r)$ .

Depending on  $p, \rho, \ell$  and  $r$ ,  $\mathcal{DS}_\rho(p, \ell, r)$  is either empty (if  $p$  is in the ellipse) or it is bounded by two rays and an elliptic arc; see Figure 3. In that case, it can also be interpreted as an elliptic arc and an interval of polar angles around  $p$ .



■ **Figure 3** A dead sector  $\mathcal{DS}_\rho(p, \ell, r)$ , shaded in gray and red with the ellipse  $E(\ell, r, \rho)$ . We approximate the ellipse by a disk centered at  $p$  with radius  $\tilde{A}_\rho(p, \ell, r)$  (thereby ignoring the red area), which is the distance from  $p$  to the farthest point  $q$  of the rectangle  $B(\ell, r, \rho)$  in  $\mathcal{DS}_\rho(p, \ell, r)$ .

During our enumeration we construct many dead sectors, the union of which can become quite complex, making it cumbersome and inefficient to work with directly. We instead chose to simplify the shape of our dead sectors, giving up a small fraction of excluded area in exchange for a simple and efficient representation. We replace the elliptic arc by a single disk centered on  $p$ , whose radius is at least the maximum distance from  $p$  to any point on the elliptic arc. We can hence represent each non-empty simplified dead sector by a polar angle interval around  $p$  and a single radius called *activation distance*  $A_\rho(p, \ell, r)$ ; see Figure 3.

We initially attempted to compute fairly precise upper bounds on the activation distance; however, due to computational and numerical issues described in the full version, we decided to use a more robust and efficient approach. Instead of using the elliptic arc, we compute an upper bound  $\tilde{A}_\rho(p, \ell, r)$  using a minimal rectangle  $B(\ell, r, \rho)$  containing the ellipse with sides are parallel and perpendicular to  $\ell r$ . We only need to check the extreme points of  $B(\ell, r, \rho)$  and its intersections with the rays  $L(p, \ell, r)$  and  $R(p, \ell, r)$  to compute an upper bound  $\tilde{A}_\rho(p, \ell, r)$ ; see Figure 3.

In the worst case, these simplifications may lead to additional candidate edges. While this cannot make the resulting graph exclude any edges that satisfy the ellipse property, it is still undesirable; we use additional checks for further reductions later on.

### 3.4 Quadtree

We use a quadtree containing all points in  $P$  for the filtered incremental search. The points are stored in a contiguous array  $A_P$  outside the tree. Each quadtree node  $v$  is associated with a contiguous subrange of  $A_P$  represented by two pointers. This subrange contains the points in the subtree  $\mathcal{T}_v$  rooted at  $v$ . Each node also has a bounding box that contains all points in  $\mathcal{T}_v$ . Each interior node has precisely four children; each leaf node contains at most a small constant number of points. To allow the contiguity of the subranges, points are reordered during tree construction. This has the added benefit of spatially sorting the points, improving the probability that geometrically close points are near each other in memory [29].

### 3.5 Enumeration process

One can think of the filtered incremental search from  $p$  as a process of continuously growing a disk centered at  $p$  starting with a radius of 0. As in the sweep-line paradigm, one encounters different types of events at discrete disk radii. We primarily encounter events when the disk first touches a point of  $P$  or the bounding box of a quadtree node.

Observe that, during a search from a point  $p$ , the dead sectors have two states: either their activation distance is not yet reached, in which case they are *inactive* and do not exclude any points, or they are *active* and exclude all points in a certain polar angle interval around  $p$ . We therefore also introduce events when the disk radius reaches the activation distance of a dead sector. This enables efficient management of active dead sectors as a set of polar angle intervals around  $p$ ; see the full version for a discussion of the ensuing numerical issues.

When we first encounter a point  $t$ , we have to determine whether  $t$  is in any dead sector by checking the active dead sector data structure. If it is not, we have to report it as potential neighbor of  $p$ , adding it to a set of points sorted by polar angle around  $p$ . Furthermore, to construct new dead sectors, we combine  $t$  with  $O(1)$  other points of  $P$ ; which points we use is decided by a heuristic discussed in the full version.

When we encounter a quadtree node  $v$ , we have to determine whether  $v$ 's bounding box is fully contained in the union of all dead sectors and can thus be pruned; we thus again check the active dead sectors. Otherwise, we have to take  $v$ 's children, or the points it contains if it is a leaf, into account; they are then considered as future events.

### 3.6 Initialization and postprocessing

We initially compute the Delaunay triangulation of the point set  $P$  and compute its dilation. We also optionally attempt to improve the dilation of the triangulation by a simple improvement heuristic. The heuristic is based on computing constrained Delaunay triangulations, greedily adding shortcut edges as constraints to reduce the length of the path currently defining the dilation. We then use the resulting dilation  $\rho$  as bound for the enumeration process outlined in the previous sections, enumerating only edges that could locally be in a triangulation with dilation strictly below  $\rho$ .

After the initial enumeration process is complete, we are left with a set of *possible* edges and can safely ignore all other edges. We postprocess these as follows. For each possible edge  $pq$ , we compute the set of possible edges intersecting it. We need this information later on to model the problem of finding a triangulation on the set of possible edges. For each pair  $st, \ell r$  of intersecting edges, we explicitly check whether either pair is an exclusion certificate for the other. In many cases, this postprocessing gets us very close to the edge set that would be obtained by the trivial  $\Theta(n^4)$  edge candidate enumeration algorithm; see the experiment section for details. We mark each edge that does not have intersecting possible edges as *certain*; certain edges must be part of any triangulation with dilation less than  $\rho$ .

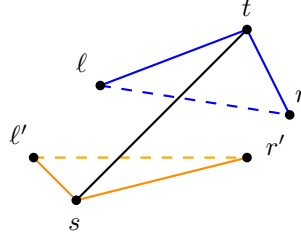
### 3.7 Dilation thresholds

We also compute a *dilation threshold*  $\vartheta(st)$  for each possible edge  $st$ . Let  $I(st)$  be the set of possible edges intersecting  $st$ . For each  $\ell r \in I(st)$ , we can compute a lower bound on the dilation of the shortest path connecting  $\ell$  to  $r$ , should  $st$  be present, as

$$\rho_{st}(\ell r) = \min\{d(\ell, s) + d(s, r), d(\ell, t) + d(t, r)\} / d(\ell, r);$$

see Figure 4. The dilation threshold of  $st$  is then  $\vartheta(st) = \max_{\ell r \in I(st)} \rho_{st}(\ell r)$ .





■ **Figure 4** If  $st$  (black) is present, the pairs  $\ell, r$  (resp.  $\ell', r'$ ) can only be connected by paths that have at least the length of the solid blue (resp. orange) path. Hence, the ratio between the solid and dashed blue (resp. orange) paths is a lower bound on the dilation of any triangulation containing  $st$ . The maximum of such lower bounds for  $st$  is the dilation threshold  $\vartheta(st)$  of  $st$ .

► **Observation 4.** *If an edge  $st$  has dilation threshold  $\vartheta(st)$ , it is not in any triangulation with dilation  $\rho < \vartheta(st)$ .*

This allows us to quickly exclude further edges if we lower the dilation bound  $\rho$  we aim for, without reenumerating edges or recomputing intersections.

### 3.8 Lower bounds

We use the dilation thresholds to bound the minimum dilation. For each edge  $st$ , either  $st$  or some edge crossing it must be in any triangulation. Thus, the minimum of all dilation thresholds among  $\{st\} \cup I(st)$  is a lower bound on the minimum dilation. Combining all edges, we obtain the following lower bound:

$$\rho(T) \geq \max_{st} \min\{\vartheta(pq) \mid pq \in \{st\} \cup I(st)\}.$$

We use interval arithmetic to compute a safe lower bound on this value in  $O(1)$  time per intersection between two edges resulting from our enumeration.

Furthermore, by computing the points on the convex hull, we also know the number of edges  $g$  of any triangulation. This also gives us a lower bound on the minimum dilation by considering the  $g$  lowest dilation thresholds. We also use Kruskal's algorithm to compute the lowest dilation threshold that admits a connected graph.

## 4 Exact algorithms

Now we present two exact algorithms: INCMDT is an incremental method that uses a SAT solver for iterative improvement, until it can prove that no better solution exists. BINMDT is based on a binary search for the optimal dilation  $\rho$ ; once the lower and upper bound are reasonably close, the approach falls back to INCMDT to reach a provably optimal solution.

### 4.1 Triangulation supergraph

Both algorithms rely on the MDT supergraph mentioned in Section 3. As part of this computation, we also obtain an initial triangulation and its dilation, as well the intersecting possible edges  $I(st)$  for each possible edge  $st$ . In both algorithms, we may gradually discover triangulations with lower dilation; these are used to exclude additional edges using the precomputed dilation thresholds  $\vartheta(e)$ . To keep track of the status of each edge, we insert all points and possible edges into a graph data structure we call *triangulation supergraph*. In



this structure, we mark each edge as *possible*, *impossible* or *certain*. Initially, all enumerated edges are *possible*. If, at any point, all edges intersecting an edge  $e$  become *impossible*,  $e$  becomes *certain*. If an *impossible* edge becomes *certain* or vice versa, the graph does not contain a triangulation any longer. If this happens, we say we encounter an *edge conflict*.

## 4.2 SAT formulation

Given a triangulation supergraph  $G = (P, E)$ , we model the problem of finding a triangulation on *possible* and *certain* edges using the following simple SAT formulation. Let  $E_p \subseteq E$  be the set of non-*impossible* edges when the SAT formulation is constructed. For each edge  $e \in E_p$ , we have a variable  $x_e$ . We use the following clauses in our formulation.

$$\neg x_{e_1} \vee \neg x_{e_2} \quad \forall e_1, e_2 \in E_p : e_2 \in I(e_1) \quad (1)$$

$$x_e \vee \bigvee_{e_j \in I(e)} x_{e_j} \quad \forall e \in E_p \quad (2)$$

Clauses (1) ensure crossing-freeness and clauses (2) ensure maximality. When an edge  $e$  becomes *certain*, we add the clause  $x_e$ ; similarly, when an edge becomes *impossible*, we add the clause  $\neg x_e$ . Both algorithms are based on this simple formulation; in the following, we describe how they use and modify it to find an MDT.

## 4.3 Clause generation

The following subproblem, which we call *dilation path separation*, arises in both our algorithms: Given a dilation bound  $\rho$ , a triangulation supergraph  $G = (P, E)$  excluding only edges that cannot be in any triangulation with dilation less than  $\rho$ , a current triangulation  $T$  and a pair of points  $s, t \in P$  such that  $|\pi_T(s, t)| \geq \rho \cdot d(s, t)$ , find a clause  $C$  that is (a) violated by  $T$  and (b) satisfied by any triangulation  $T'$  with  $\rho(T') < \rho$ .

► **Lemma 5.** *Assuming a polynomial-time oracle for comparing sums of square roots, there is a polynomial-time algorithm that solves the dilation path separation problem.*

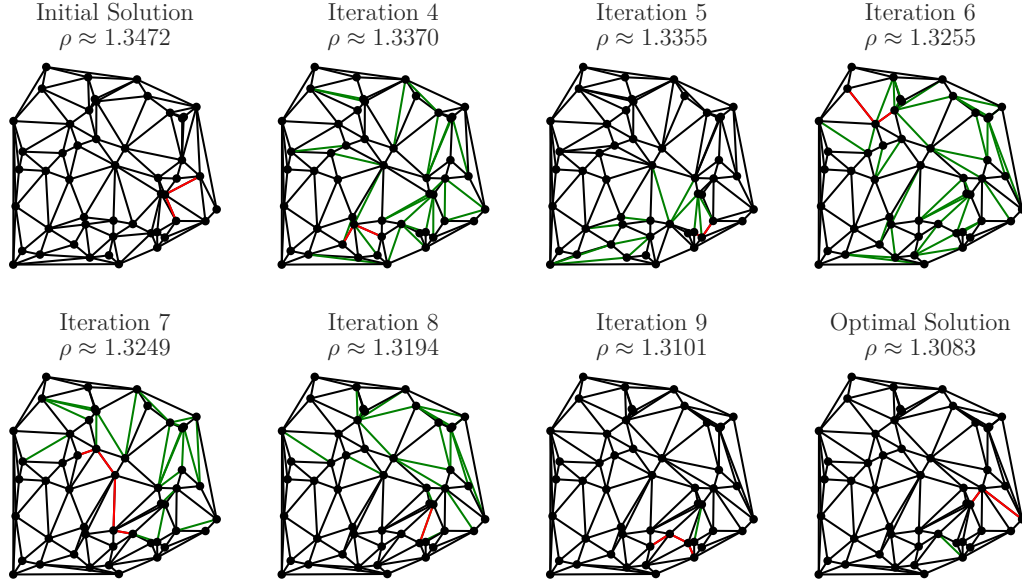
**Proof.** Let  $\Pi$  be the set of all  $s$ - $t$ -paths  $\pi$  in  $G$  with  $|\pi| < \rho \cdot d(s, t)$ . We begin by observing that, along every path  $\pi \in \Pi$ , there is an edge  $e \in E$  that is not in  $T$ ; otherwise, we get a contradiction to  $|\pi_T(s, t)| \geq \rho \cdot d(s, t)$ . Let  $E' \subseteq E \setminus T$  be a set of edges such that for each  $\pi \in \Pi$ , there is an edge  $e \in E'$  on  $\pi$ . Then,  $C = \bigvee_{e \in E'} x_e$  is a clause that satisfies the requirements; note that if  $\Pi$  is empty, the empty clause can be returned.

$T$  contains no edge from  $E'$ , so  $C$  is violated by  $T$ . Furthermore, if a triangulation  $T'$  with  $\rho(T') < \rho$  does not contain any of the edges in  $E'$ , it contains none of the paths in  $\Pi$ . Therefore,  $\pi_{T'}(s, t)$  uses an edge that is not in  $E$ , which has been excluded from all triangulations with dilation less than  $\rho$ ; a contradiction.  $E'$  can be computed by repeatedly computing shortest  $s$ - $t$ -paths  $\pi$ ; as long as  $\pi < \rho d(s, t)$ , we find an edge  $e \notin T$  on  $\pi$ , add  $e$  to  $E'$  and forbid it in future paths. The number of edges bounds the number of iterations of this process; using the comparison oracle, we can efficiently perform each iteration. ◀

For a description of how we compute  $E'$  in practice, see the full version.

## 4.4 Incremental algorithm

Based on the SAT formulation and the algorithm for the dilation path separation problem, INCMDT is simple. Given an initial triangulation  $T$  with dilation  $\rho$ , we enumerate the set of candidate edges and construct a triangulation supergraph  $G$  with bound  $\rho$ . We construct the



■ **Figure 5** Progress of the incremental algorithm on an instance with  $n = 50$  points. Green edges indicate changes in the triangulation, red edges indicate a dilation-defining path.

initial SAT formula  $M$  and solve it; if it is unsatisfiable, the initial triangulation is optimal. Otherwise, we repeat the following until the model becomes unsatisfiable or we encounter an edge conflict, keeping track of the best triangulation found, see Figure 5.

We extract the new triangulation  $T'$  from the SAT solver and compute the dilation  $\rho'$  and a pair  $s, t$  of points realizing  $\rho'$ . If  $\rho'$  is better than the best previously found dilation  $\rho$ , we update  $\rho$  and mark all edges  $e$  with  $\vartheta(e) \geq \rho'$  as *impossible*. We then set  $T = T'$  and solve the dilation path separation problem for  $\rho$ ,  $G$ ,  $T$ ,  $s$  and  $t$ . We add the resulting clause to  $M$  and let the SAT solver find a new solution.

## 4.5 Binary search

Preliminary experiments with INCMDT showed that we spend almost all runtime for computing dilations, even for instances for which we could rely exclusively on interval arithmetic, requiring no exact computations. For many instances, most iterations of INCMDT resulted in tiny improvements of the dilation. To reduce the number of iterations (and thus, dilations computed), we considered the binary search-based algorithm BINMDT.

### 4.5.1 High-level idea

At any point in time, aside from the dilation  $\rho_{\text{ub}}$  of the best known triangulation, BINMDT maintains a lower bound  $\rho_{\text{lb}}$  on the dilation, initialized as described in Section 3.

As long as  $\rho_{\text{ub}} - \rho_{\text{lb}} \geq \sigma$  for a small threshold value  $\sigma$ , BINMDT performs a binary search. It computes a new dilation bound  $\rho = \frac{1}{2}(\rho_{\text{lb}} + \rho_{\text{ub}})$ . It then uses the SAT model in a similar way as INCMDT to determine whether a triangulation  $T$  with  $\rho(T) < \rho$  exists. If it does, it updates  $\rho_{\text{ub}} = \rho(T)$ ; otherwise, it updates  $\rho_{\text{lb}} = \rho$ .

Once  $\rho_{\text{ub}} - \rho_{\text{lb}}$  falls below  $\sigma$ , BINMDT falls back to a slightly modified version of INCMDT to find the MDT, starting from the best known triangulation with dilation  $\rho_{\text{ub}}$ .

In the following, we describe and motivate the differences between how INCMDT and BINMDT use the SAT formulation; for more details, see the full version.

### 4.5.2 Dilation sampling

To further reduce the time spent on computing dilations, observe the following. When a node  $v$  of some graph  $G$  is expanded in Dijkstra's algorithm from source  $s$ , we know the shortest path from  $s$  to  $v$  and thus the dilation  $\rho_G(s, v)$ . Because  $\rho_G(s, v) \leq \rho(G)$ , we can compute a lower bound on the dilation much faster than the precise value by only performing a constant number of node expansions from each point  $p \in P$ . We call this *sampling* of the dilation. Given a bound  $\rho$  on the dilation, we can sample a triangulation  $T$  for violations, i.e., pairs  $s, t$  of points with  $\rho_T(s, t) \geq \rho$ . We observed that a dilation-defining path usually consisted of few edges; thus, we have a good chance of finding it by sampling.

If it is likely that a new-found triangulation  $T'$  violates a given bound  $\rho$ , we can thus expect to save time by sampling for violations instead of computing the dilation exactly. Sampling also allows us to use multiple violations to construct multiple clauses in each iteration, potentially further reducing the number of iterations. BINMDT uses sampling after each SAT call with a small constant limit on the number of violations. If violations are found, no full dilation computation is required and violations are used to construct clauses. Only if no violations are found, we compute the exact dilation; ideally, this only happens once for each upper bound reduction in the binary search, namely once we find a triangulation satisfying the current bound. We also sample in the final improvement phase of BINMDT. For an experimental overview on the number of times sampling was sufficient in comparison to the number of times the dilation had to be computed exactly, see the full version.

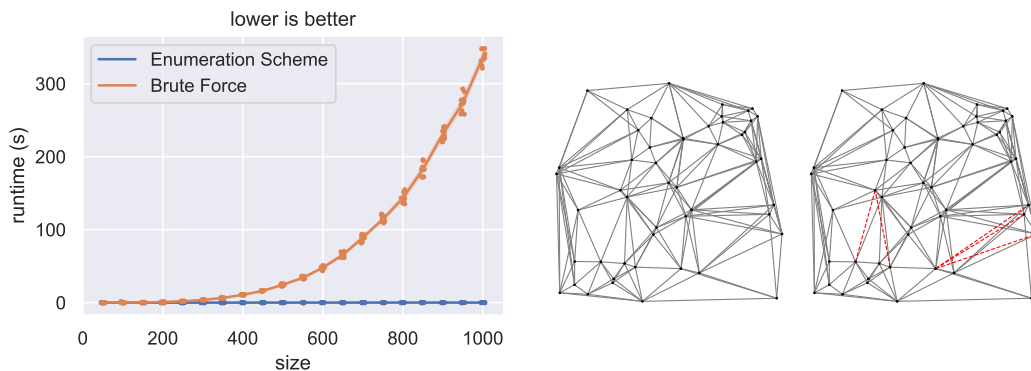
## 5 Empirical evaluation

Now we present experiments to evaluate our algorithms. We used Python 3.12, with a core module written in C++20 for all computationally heavy tasks; the code was compiled with GCC 13.2.0 in release mode. We use CGAL 5.6.1 for geometric primitives and exact number types, Boost 1.83 for utility functions and pybind11 2.12 for Python bindings and use the incremental SAT solver CaDiCaL 1.9.5 via the PySAT interface for solving the SAT models. All experiments were performed on Linux workstations equipped with AMD Ryzen 9 7900 CPUs with 12 cores/24 threads and 96 GiB of DDR5-5600 RAM running Ubuntu 24.04.1.

### 5.1 Research questions

Our experimental evaluation aims to answer the following questions.

- Q1** How does the edge enumeration algorithm from Section 3 compare to the brute force enumeration in terms of runtime and the number of edges eliminated? Can we solely rely on this algorithm or should we reconsider using the brute force enumeration?
- Q2** What is the quality of the initial solutions computed by the Delaunay triangulation and the constrained Delaunay triangulations from Section 3 compared to the optimal solution?
- Q3** How do our approaches compare to existing algorithms for the MDT w.r.t. runtime and solution quality? Can we solve instances with thousands of points to provable optimality?
- Q4** How do BINMDT and INCMDT compare? Which should be used for large instances?



■ **Figure 6** Comparison of our approach from Section 3 and the  $\Theta(n^4)$  brute force elimination on the *random* instance set. **(Left)** The  $\Theta(n^4)$  elimination is infeasible for larger instances. **(Right)** In all cases, only a small number of (red) edges were not eliminated by our approach.

## 5.2 Experiment design

To answer our questions, we collected and generated a large set of instances, consisting of instances from the following instance classes. In all cases, the coordinates of points in the instances are either integers or double precision floating-point numbers.

**random-small** We include instances from the work of [8]. The 210 instances have fixed sizes  $n \in \{10, 20, \dots, 70\}$  and were generated by placing uniformly random points inside a  $10 \times 10$  square. For each size a total of 30 instances were generated.

**random** We include two sets of randomly generated instances. These encompass a set of instances with points with float coordinates chosen uniformly between 0 and  $10^3$ , ranging from 50 to 10,000 points. This resulted in a total of 800 instances.

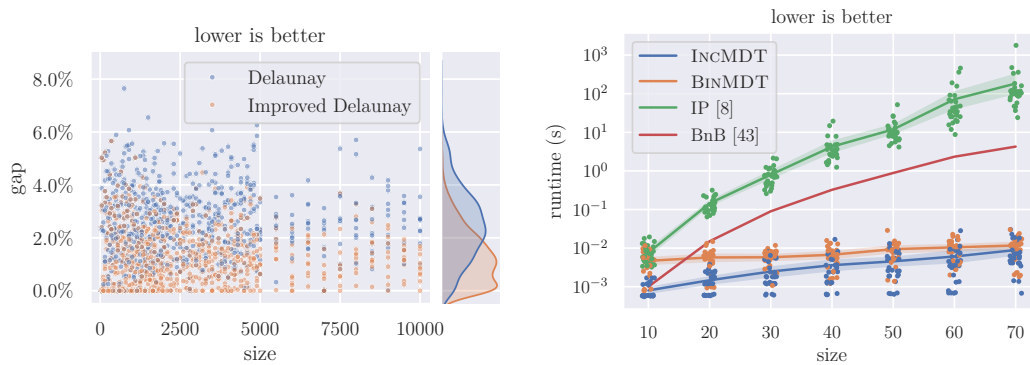
**public** We include instances from all well-known publicly available benchmark instance sets we could locate. These include point sets previously used in the CG:SHOP challenges [17, 16], TSPLIB instances [40], instances from a VLSI dataset<sup>1</sup> and point sets from the Salzburg Database of Polygonal Inputs [22]. In total, we collected 486 instances with up to 10,000 points and an additional 38 with up to 30,000 points.

## 5.3 Q1: Edge enumeration

We first compare the edge enumeration algorithm from Section 3 to the  $\Theta(n^4)$  brute force enumeration of all possible edges on the *random* instance set. Both preprocessing options include finding all pairwise intersections between the enumerated segments. Due to the  $\Theta(n^4)$  runtime, we only consider instances with up to 1000 points; see Figure 6.

The  $\Theta(n^4)$  algorithm precisely identifies all edges that have the ellipse property; it can hence only eliminate more edges than the edge enumeration algorithm from Section 3. However, for our test instances, the number of edges that can be eliminated by our approach is almost identical to the  $\Theta(n^4)$  algorithm; see Figure 6 for an example. The runtime makes the  $\Theta(n^4)$  algorithm infeasible for larger instances; it takes more than 250s for instances with only 1000 points, compared to  $< 1$ s for our more efficient approach.

<sup>1</sup> <https://www.math.uwaterloo.ca/tsp/vlsi/index.html>



■ **Figure 7 (Left)** Initial solution comparison on the *random* set with up to 10,000 points. Delaunay triangulations can be improved by shortcut edges, but are often close to the optimal solution. **(Right)** Runtime comparison with the approaches from [8] and [41] on the *random-small* set.

## 5.4 Q2: Initial solutions

Better initial solutions can reduce the number of candidate edges further than bad ones and thus affect the runtime of the overall algorithm. Delaunay triangulations are a natural choice for the initial solution, but we suspect that they can be improved by adding shortcut edges. Figure 7 shows that the relative dilation gap between the Delaunay triangulation and the optimal solution for the *random* instance set. It can be seen that the introduction of shortcut edges can indeed reduce the gap to the MDT to around 1.5 %.

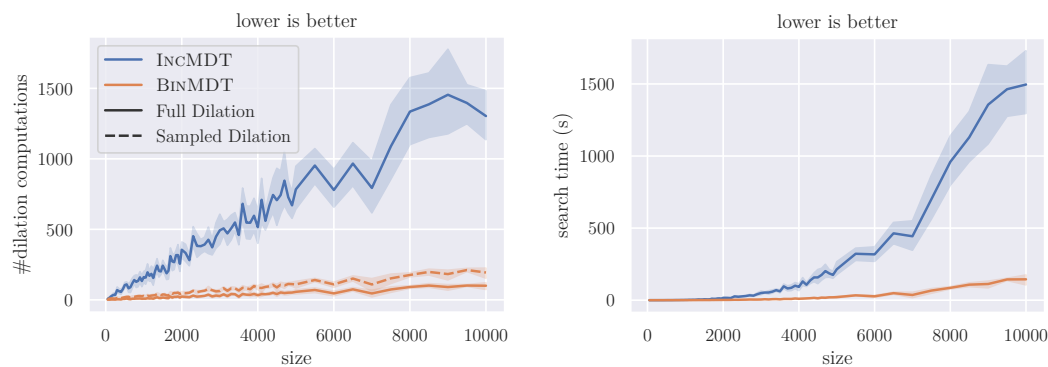
## 5.5 Q3: Comparison to state-of-the-art

We compare our approaches to two exact state-of-the-art algorithms for the MDT. Note that both of these use floating-point arithmetic and are not guaranteed to find the optimal solution. However, we can confirm that all previous solutions are within a small relative error of our optimal solution. The first approach is an integer programming (IP) approach by [8] that used the commercial software CPLEX to solve the MDT. The second comparison is with the most recent exact algorithm (BnB) from Sattari and Izadi [41]. For both approaches the source code is no longer available, so we cannot compare our results on the same hardware. Also for BnB [41] the instance data and results are no longer available. For both approaches we therefore use the data the authors published in their papers. Note that the drastic runtime difference that we see in our experiments cannot be attributed to improved hardware alone.

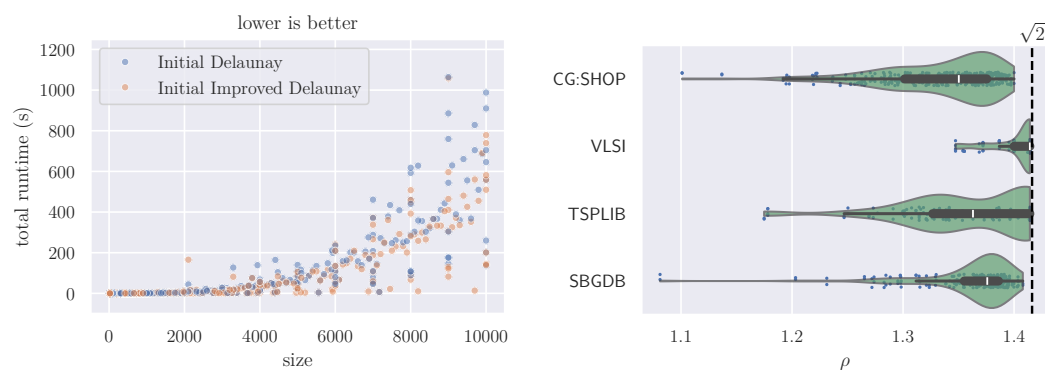
For *random-small*, both INCMDT and BINMDT outperform the IP and BnB approach by a large margin (up to four orders of magnitude), see Figure 7. All instances were solvable in less than 0.1 s. Additionally, Sattari and Izadi [41] provided results for TSPLIB [40] instances (part of our *public* instance set) with up to 200 points. Our approach is significantly faster than theirs, solving each of these instances to provable optimality in less than 1 s instead of up to 1248 s; a table with all instances and runtimes can be found in the full version.

## 5.6 Q4: Algorithm comparison

We now compare INCMDT to BINMDT. Recall that BINMDT aims to reduce the time spent on dilation computations by reducing the number of dilation computations and merely sampling the dilation whenever that suffices. The first experiment was conducted on the



**Figure 8** Experiments on the *random* benchmark set. **(Left)** The number of dilation computations of BINMDT is significantly reduced compared to the incremental algorithm. **(Right)** BINMDT is significantly faster than the incremental algorithm.



**Figure 9** Experiments on the *public* benchmark set. **(Left)** Using the improved Delaunay triangulation as an initial solution significantly improved the performance. **(Right)** The dilation of the MDTs is at most  $\sqrt{2}$  for all instances.

*random* instances with up to 10,000 points; this experiment confirms that BINMDT indeed achieves a significantly lower runtime, requires fewer dilation computations, and that sampling is sufficient in most cases, see Figure 8.

After we confirmed that BINMDT is superior, we conducted an additional experiment with two different initial solution strategies on the *public* instances up to 10,000 points, see Figure 9. The first strategy uses the Delaunay triangulation as an initial solution; the second strategy uses the improved Delaunay triangulation with shortcut edges. The improved Delaunay triangulation significantly reduces the runtime of BINMDT for almost all instances. Detailed results for all *public* instances, as well as an additional experiment on the *public* instance set showing that BINMDT can solve instances with up to 30,000 points in less than 17 h to provable optimality, can be found in the full version.

## 6 Improved bounds for regular $n$ -gons

Vertex sets of regular  $n$ -gons are particularly challenging, as there are no points in the interior to serve as Steiner points for shortcuts, making local heuristics less successful. They are also natural candidates for worst-case dilation, as each diagonal constitutes an obstacle for the

separated points. Thus, they have received considerable attention [35, 20, 43], with a lower bound of 1.4308 (see [20]) and an upper bound of 1.4482 (see [43]) on their worst-case dilation. Improving this gap is an open question posed by [20, Problem 1], originating from [7, 30]. With the help of our exact algorithm (see the full version), we were able to compute bounds for  $n \in \{4, 5, \dots, 100\}$ , answering the problem by Dumitrescu and Ghosh.

Our implementation currently uses floating-point precision to represent the coordinates of the input points. Thus, we cannot exactly represent the necessarily irrational points of any regular  $n$ -gon ( $n \geq 5$ ) in our solver. However, we can compute the MDT of a rational point set that is a good approximation of a regular  $n$ -gon. We can then bound the error and thus obtain a rigorous lower bound on the dilation of the regular  $n$ -gon.

► **Theorem 6.** *Let  $P$  be a set of  $n = 84$  points placed at the vertices of a regular  $n$ -gon. Then the dilation of the MDT of  $P$  is  $\rho \geq 1.44116645381$ .*

**Proof.** Let  $P$  be the point set of a regular 84-gon and  $Q$  be a point set that contains floating-point approximations of the points of  $P$ . There is a bijection  $f : P \rightarrow Q$  with inverse  $f^{-1}$  that maps each point in  $P$  to the closest point in  $Q$ . For a given triangulation or path of  $P$ , we write  $f(\cdot)$  to denote the triangulation or path where the points are transformed by pointwise application of  $f$ . Neither  $P$  nor  $Q$  contain collinear points and all points are in convex position, therefore  $T$  is a triangulation of  $P$  iff  $f(T)$  is a triangulation of  $Q$ .

Let  $\epsilon \geq \max_{p_i, p_j \in P} |d(p_i, p_j) - d(f(p_i), f(p_j))|$  be a bound on the maximum absolute error on distances and let  $\delta$  be a chosen such that

$$\forall p_i, p_j \in P : (1 - \delta)d(p_i, p_j) \leq d(f(p_i), f(p_j)) \leq (1 + \delta)d(p_i, p_j).$$

Let  $T$  be the MDT of  $P$  with dilation  $\rho$ . Let  $f(p_i), f(p_j)$  be a dilation-defining pair in  $f(T)$  with path  $\pi \subset Q$ . Because  $T$  has dilation  $\rho$ , we have (i)  $|f^{-1}(\pi)|/d(p_i, p_j) \leq \rho$ . As  $\pi$  has at most  $n - 1$  edges, we have (ii)  $|\pi| - (n - 1)\epsilon \leq |f^{-1}(\pi)| \leq |\pi| + (n - 1)\epsilon$  and thus the following upper bound on the MDT of  $Q$ .

$$\begin{aligned} \frac{|\pi|}{d(f(p_i), f(p_j))} &= \frac{|\pi| - (n - 1)\epsilon}{d(f(p_i), f(p_j))} + \frac{(n - 1)\epsilon}{d(f(p_i), f(p_j))} \stackrel{(ii)}{\leq} \frac{|f^{-1}(\pi)|}{(1 - \delta)d(p_i, p_j)} + \frac{((n - 1)\epsilon)}{d(f(p_i), f(p_j))} \\ &\stackrel{(i)}{\leq} \frac{1}{1 - \delta}\rho + \frac{(n - 1)\epsilon}{d(f(p_i), f(p_j))} \leq \frac{1}{1 - \delta}\rho + \frac{(n - 1)\epsilon}{\min_{q_i, q_j \in Q} d(q_i, q_j)}. \end{aligned}$$

Using exact calculations done with *sympy*, we computed  $\epsilon \leq 2.730751 \cdot 10^{-16}$  and  $\delta \leq 6.458762 \cdot 10^{-16}$ , which, combined with the lower bound on the dilation of  $Q$  from our solver gives us  $\rho \geq 1.44116645381$ . ◀

## 7 Conclusion

We have presented exact algorithms for minimum dilation triangulations, greatly outperforming previous methods from the literature. This has also yielded insights into the intricate structure of optimal solutions for regular  $n$ -gons, together with new lower bounds on the worst-case dilation of triangulations. This demonstrates the value of computational tools for gaining analytic insights that seem out of reach with purely manual analysis.

On the theoretical side, one tantalizing open question remains the complexity of the MDT; here the intricate structure of solutions for regular  $n$ -gons may show the way for hardness constructions even for convex arrangements. An equally fascinating problem is the worst-case dilation of triangulations; here we conjecture that the asymptotic dilation for regular  $n$ -gons corresponds to the worst-case dilation for general point sets, implying a value between 1.44116 and 1.4482, with a better lower bound achievable via even larger  $n$ -gons.



On the practical side, the runtime of our algorithms is dominated by repeatedly computing the dilation of a triangulation. With sums of square roots handled as described, this does not appear to be a difficult problem per se, so appropriately tailored algorithms or data structures for carrying out huge numbers of dilation queries would be extremely helpful.

---

## References

- 1 Khaled M. Alzoubi, Xiang-Yang Li, Yu Wang, Peng-Jun Wan, and Ophir Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Trans. Parallel Distributed Syst.*, 14(4):408–421, 2003. doi:10.1109/TPDS.2003.1195412.
- 2 Narayanasetty Amarnadh and Pinaki Mitra. Upper bound on dilation of triangulations of cyclic polygons. In *Computational Science and Its Applications - ICCSA 2006*, volume 3980 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2006. doi:10.1007/11751540\_1.
- 3 Boris Aronov, Mark de Berg, Otfried Cheong, Joachim Gudmundsson, Herman J. Haverkort, Michiel H. M. Smid, and Antoine Vigneron. Sparse geometric graphs with small dilation. *Comput. Geom.*, 40(3):207–219, 2008. doi:10.1016/J.COMGEO.2007.07.004.
- 4 Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean geometry*, pages 47–123. World Scientific, 1995.
- 5 Ahmad Biniiaz, Mahdi Amani, Anil Maheshwari, Michiel H. M. Smid, Prosenjit Bose, and Jean-Lou De Carufel. A plane 1.88-spanner for points in convex position. *J. Comput. Geom.*, 7(1):520–539, 2016. doi:10.20382/JOCG.V7I1A21.
- 6 Johannes Blömer. Computing sums of radicals in polynomial time. In *Symposium on Foundations of Computer Science (FOCS)*, pages 670–677, 1991. doi:10.1109/SFCS.1991.185434.
- 7 Prosenjit Bose and Michiel H. M. Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom.*, 46(7):818–830, 2013. doi:10.1016/J.COMGEO.2013.04.002.
- 8 Aléx F. Brandt, Miguel M. Gaiowski, Cid C. de Souza, and Pedro J. de Rezende. Minimum dilation triangulation: Reaching optimality efficiently. In *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014*. Carleton University, Ottawa, Canada, 2014. URL: <http://www.cccg.ca/proceedings/2014/papers/paper09.pdf>.
- 9 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, and Sampson Wong. Bicriteria approximation for minimum dilation graph augmentation. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024*, volume 308 of *LIPICs*, pages 36:1–36:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ESA.2024.36.
- 10 Leizhen Cai and Derek G. Corneil. Tree spanners. *SIAM J. Discret. Math.*, 8(3):359–387, 1995. doi:10.1137/S0895480192237403.
- 11 Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New sparseness results on graph spanners. *Int. J. Comput. Geom. Appl.*, 5:125–144, 1995. doi:10.1142/S0218195995000088.
- 12 Otfried Cheong, Herman J. Haverkort, and Mira Lee. Computing a minimum-dilation spanning tree is NP-hard. *Comput. Geom.*, 41(3):188–205, 2008. doi:10.1016/J.COMGEO.2007.12.001.
- 13 Paul Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the Second Annual ACM SIGACT/SIGGRAPH Symposium on Computational Geometry*, pages 169–177. ACM, 1986. doi:10.1145/10515.10534.
- 14 Paul Chew. There are planar graphs almost as good as the complete graph. *J. Comput. Syst. Sci.*, 39(2):205–219, 1989. doi:10.1016/0022-0000(89)90044-5.
- 15 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications, 3rd Edition*. Springer, 2008. doi:10.1007/978-3-540-77974-2.

- 16 Erik D Demaine, Sándor P Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing convex partitions for point sets in the plane: The CG:SHOP Challenge 2020. *arXiv preprint*, 2020. [arXiv:2004.04207](https://arxiv.org/abs/2004.04207).
- 17 Erik D Demaine, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Area-optimal simple polygonalizations: The CG Challenge 2019. *Journal of Experimental Algorithmics (JEA)*, 27(2):1–12, 2022.
- 18 Erik D. Demaine, Joseph S. B. Mitchell, and Joseph O’Rourke. The open problems project: Problem #33, 2001. URL: <http://topp.openproblem.net/>.
- 19 Maria Gisela Dorzán, Mario Guillermo Leguizamón, Efrén Mezura-Montes, and Gregorio Hernández-Peñalver. Approximated algorithms for the minimum dilation triangulation problem. *J. Heuristics*, 20(2):189–209, 2014. doi:10.1007/S10732-014-9237-2.
- 20 Adrian Dumitrescu and Anirban Ghosh. Lower bounds on the dilation of plane spanners. *Int. J. Comput. Geom. Appl.*, 26(2):89–110, 2016. doi:10.1142/S0218195916500059.
- 21 Herbert Edelsbrunner and Tiow Seng Tan. A quadratic time algorithm for the minimax length triangulation. *SIAM J. Comput.*, 22(3):527–551, 1993. doi:10.1137/0222036.
- 22 Günther Eder, Martin Held, Steinþór Jasonarson, Philipp Mayer, and Peter Palfrader. Salzburg database of polygonal data: Polygons and their generators. *Data in Brief*, 31:105984, 2020. doi:10.1016/j.dib.2020.105984.
- 23 Friedrich Eisenbrand, Matthieu Haeberle, and Neta Singer. An improved bound on sums of square roots via the subspace theorem. In *Symposium on Computational Geometry (SoCG)*, 2024.
- 24 David Eppstein. Spanning trees and spanners. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. North Holland / Elsevier, 2000. doi:10.1016/B978-044482537-7/50010-3.
- 25 Kai-Wei Fan, Sha Liu, and Prasun Sinha. Scalable data aggregation for dynamic events in sensor networks. In Andrew T. Campbell, Philippe Bonnet, and John S. Heidemann, editors, *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys 2006*, pages 181–194. ACM, 2006. doi:10.1145/1182807.1182826.
- 26 Arthur M. Farley, Andrzej Proskurowski, Daniel Zappala, and Kurt J. Windisch. Spanners and message distribution in networks. *Discret. Appl. Math.*, 137(2):159–171, 2004. doi:10.1016/S0166-218X(03)00259-2.
- 27 Sándor P. Fekete, Winfried Hellmann, Michael Hemmer, Arne Schmidt, and Julian Troegel. Computing maxmin edge length triangulations. *J. Comput. Geom.*, 9(1):1–26, 2018. doi:10.20382/JOCG.V9I1A1.
- 28 Panos Giannopoulos, Rolf Klein, Christian Knauer, Martin Kutz, and Dániel Marx. Computing geometric minimum-dilation graphs is NP-hard. *Int. J. Comput. Geom. Appl.*, 20(2):147–173, 2010. doi:10.1142/S0218195910003244.
- 29 Andreas Haas. Solving large-scale minimum-weight triangulation instances to provable optimality. In *34th International Symposium on Computational Geometry, SoCG 2018*, volume 99 of *LIPICs*, pages 44:1–44:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.SOCG.2018.44.
- 30 Iyad Kanj. Geometric spanners: Recent results and open directions. In *Third International Conference on Communications and Information Technology, ICCIT 2013*, pages 78–82. IEEE, 2013. doi:10.1109/ICCITECHNOLOGY.2013.6579526.
- 31 Alexander Klein. Effiziente Berechnung einer dilationsminimalen Triangulierung. Master’s thesis, Diplomarbeit, Rheinische Friedrich-Wilhelms-Universität Bonn, 2006.
- 32 Christian Knauer and Wolfgang Mulzer. An exclusion region for minimum dilation triangulations. In *(Informal) Proceedings of the 21st European Workshop on Computational Geometry (EuroCG 2005)*, pages 33–36. Technische Universiteit Eindhoven, 2005. URL: <http://www.win.tue.nl/EWCG2005/Proceedings/9.pdf>.
- 33 László Kozma. Minimum average distance triangulations. In Leah Epstein and Paolo Ferragina, editors, *20th Annual European Symposium on Algorithms (ESA 2012)*, volume 7501 of *Lecture Notes in Computer Science*, pages 695–706. Springer, 2012. doi:10.1007/978-3-642-33090-2\_60.

- 34 Joseph SB Mitchell and Wolfgang Mulzer. Proximity algorithms. In *Handbook of Discrete and Computational Geometry*, pages 849–874. CRC Press, Inc., 2017.
- 35 Wolfgang Mulzer. Minimum dilation triangulations for the regular n-gon. *Master’s Thesis. Freie Universität Berlin, Germany.*, 2004.
- 36 Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *J. ACM*, 55(2):11:1–11:29, 2008. doi:10.1145/1346330.1346336.
- 37 Giri Narasimhan and Michiel H. M. Smid. Approximating the stretch factor of euclidean graphs. *SIAM J. Comput.*, 30(3):978–989, 2000. doi:10.1137/S0097539799361671.
- 38 Giri Narasimhan and Michiel H. M. Smid. *Geometric spanner networks*. Cambridge University Press, 2007.
- 39 Joseph O’Rourke. Advanced problem 6369. *Amer. Math. Monthly*, 88(10):769, 1981.
- 40 G. Reinelt. TSPLIB—A Traveling Salesman Problem Library. *ORSA Journal of Computing*, 3(4):376–384, 1991. doi:10.1287/IJOC.3.4.376.
- 41 Sattar Sattari and Mohammad Izadi. An exact algorithm for the minimum dilation triangulation problem. *J. Glob. Optim.*, 69(2):343–367, 2017. doi:10.1007/S10898-017-0517-X.
- 42 Sattar Sattari and Mohammad Izadi. Upper bounds for minimum dilation triangulation in two special cases. *Inf. Process. Lett.*, 133:33–38, 2018. doi:10.1016/J.IPL.2018.01.001.
- 43 Sattar Sattari and Mohammad Izadi. An improved upper bound on dilation of regular polygons. *Comput. Geom.*, 80:53–68, 2019. doi:10.1016/J.COMGEO.2019.01.009.
- 44 FNU Shariful, Justin Weathers, Anirban Ghosh, and Giri Narasimhan. Engineering an algorithm for constructing low-stretch geometric graphs with near-greedy average-degrees. *CoRR*, 2023. doi:10.48550/arXiv.2305.11312.
- 45 Israel Vite Silva, Nareli Cruz Cortés, Gregorio Toscano Pulido, and Luis Gerardo de la Fraga. Optimal triangulation in 3D computer vision using a multi-objective evolutionary algorithm. In *Applications of Evolutionary Computing, EvoWorkshops 2007*, volume 4448 of *Lecture Notes in Computer Science*, pages 330–339. Springer, 2007. doi:10.1007/978-3-540-71805-5\_36.
- 46 Victor J. D. Tsai. Delaunay triangulations in TIN creation: An overview and a linear-time algorithm. *Int. J. Geogr. Inf. Sci.*, 7(6):501–524, 1993. doi:10.1080/02693799308901979.
- 47 Chun-Hsien Wu, Kuo-Chuan Lee, and Yeh-Ching Chung. A delaunay triangulation based method for wireless sensor network deployment. *Comput. Commun.*, 30(14-15):2744–2752, 2007. doi:10.1016/J.COMCOM.2007.05.017.
- 48 Ge Xia. The stretch factor of the delaunay triangulation is less than 1.998. *SIAM J. Comput.*, 42(4):1620–1659, 2013. doi:10.1137/110832458.
- 49 Hongyu Zhou, Hongyi Wu, Su Xia, Miao Jin, and Ning Ding. A distributed triangulation algorithm for wireless sensor networks on 2d and 3d surface. In *30th IEEE International Conference on Computer Communications (INFOCOM 2011)*, pages 1053–1061. IEEE, 2011. doi:10.1109/INFOCOM.2011.5934879.