Exact Algorithms for Minimum Dilation Triangulation

Sándor P. Fekete[1], Phillip Keldenich[1], and Michael Perk[1]

1 **Department of Computer Science, TU Braunschweig**
`s.fekete@tu-bs.de, {keldenich, perk}@ibr.cs.tu-bs.de`

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. We provide practically robust methods and implementations for computing the 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.

Moreover, 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 $n = 100$.

Related Version arXiv:2502.18189

1 Introduction

Triangulating a set of points to optimize some objective is one of the classical problems in computational geometry. On the practical side, it has applications in wireless sensor networks [25, 26], mesh generation [1], computer vision [23], geographic information systems [24] and many other areas [5].

In this paper, we provide new results and insights for a previously studied, natural objective that considers triangulations as sparse structures with relatively 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 $\pi_T(s,t)$ in T and the Euclidean distance $d(s,t)$ between s and t, i.e. $\rho(T) = \max\{{|\pi_T(s,t)|}/{d(s,t)} \mid s,t \in P, s \neq t\}$. The Minimum Dilation Triangulation (MDT) problem asks for a triangulation T that minimizes the dilation $\rho(T)$, see Figure 1 for examples. 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.

Our contributions We present 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 (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. We also resolve an open problem

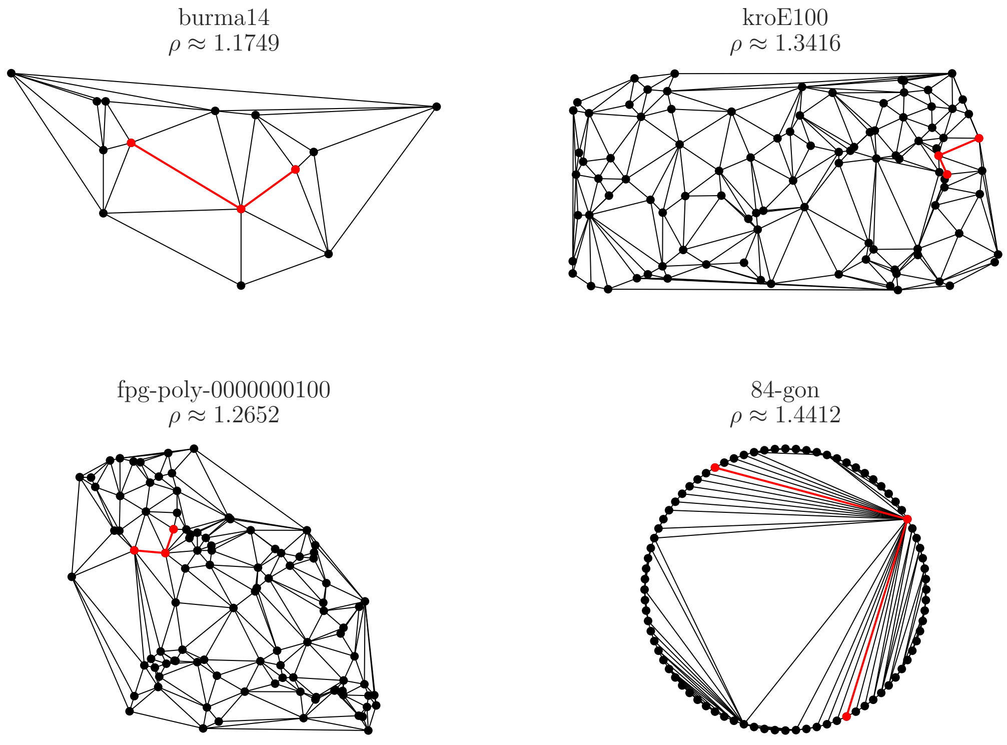


Figure 1 MDT solutions for four instances. The red edges indicate a dilation-defining path.

from Dumitrescu and Ghosh [9] by establishing a lower bound of 1.44116 on the dilation of the regular 84-gon. 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 [22]. In the process, we provide optimal solutions for regular n-gons up to $n = 100$.

Related work The complexity of finding the MDT is unknown [11, 18]. Giannopoulos et al. [12] prove that finding the minimum dilation graph with a limited number of edges is NP-hard. Cheong et al. [4] show that finding a spanning tree of given dilation is also NP-hard. Kozma [16] proves NP-hardness for minimizing the expected distance between random points in a triangulation, with edge weights instead of Euclidean distances. All practical approaches in the literature are based on fixed-precision arithmetic. Klein [14] used an enumeration algorithm to find an optimal MDT for up to 10 points. Dorzán et al. [8] 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 Brandt et al. [3] using integer linear programming techniques and the edge elimination strategy from Knauer and Mulzer [15]. Recently, Sattari and Izadi [21] presented an exact algorithm based on branch and bound that was evaluated on instances with up to 200 points.

2 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 ρ; once the lower and upper bound are reasonably close, the approach falls back to IncMDT to reach a provably optimal solution.

2.1 Enumerating possible edges

We implemented a novel and practically efficient scheme for enumerating a set of edges that induces a supergraph of the MDT with dilation *strictly less* than a given bound ρ. The approach is based on the well-known *ellipse property* (used in [3, 12, 15]) and enumerates candidate edges using a quadtree-based filtered incremental nearest-neighbor search that identifies potential neighbors while excluding points in so-called *dead sectors*. Due to space constraints, all details are deferred to the full version.

Our enumeration scheme significantly reduces the number of edges to consider, improving runtime efficiency. It also computes a dilation threshold $\vartheta(st)$ for each edge st to quickly exclude edges when lowering the dilation bound. As part of our 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.*

2.2 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.

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

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

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.

Clause generation The following subproblem, which we call *dilation path separation*, arises in both our algorithms: Given a dilation bound ρ, a triangulation supergraph $G = (P, E)$ excluding only edges that cannot be in any triangulation with dilation less than ρ, 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 $\rho(T') < \rho$. For a description of how we compute E' in practice, see the full version.

▶ **Lemma 2.1.** *Assuming a polynomial-time oracle for comparing sums of square roots, there is a polynomial-time algorithm that solves the dilation path separation problem.*

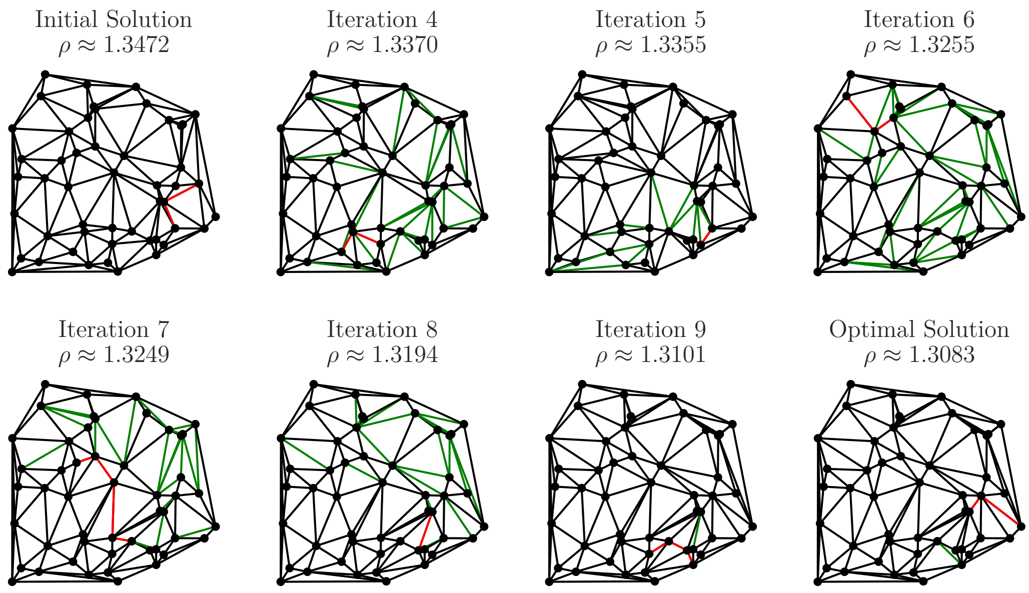


Figure 2 Progress of the incremental algorithm on an instance with $n = 50$ points. Green edges indicate changes in the triangulation, red edges indicate a dilation-defining path.

Proof. Let Π be the set of all s-t-paths π 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 π. Then, $C = \bigvee_{e \in E'} x_e$ is a clause that satisfies the requirements; note that if Π 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 Π. Therefore, $\pi_{T'}(s,t)$ uses an edge that is not in E, which has been excluded from all triangulations with dilation less than ρ; a contradiction. E' can be computed by repeatedly computing shortest s-t-paths π; as long as $\pi < \rho d(s,t)$, we find an edge $e \notin T$ on π, 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. ◀

2.3 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 ρ, we enumerate the set of candidate edges and construct a triangulation supergraph G with bound ρ. We construct the 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 2.

We extract the new triangulation T' from the SAT solver and compute the dilation ρ' and a pair s, t of points realizing ρ'. If ρ' is better than the best previously found dilation ρ, we update ρ 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 ρ, G, T, s and t. We add the resulting clause to M and let the SAT solver find a new solution.

2.4 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.

At any point in time, aside from the dilation ρ_{ub} of the best known triangulation, BinMDT maintains a lower bound ρ_{lb} on the dilation, initialized as described in the full version. As long as $\rho_{\mathrm{ub}} - \rho_{\mathrm{lb}} \geq \sigma$ for a small threshold value σ, BinMDT performs a binary search. It computes a new dilation bound $\rho = \frac{1}{2}(\rho_{\mathrm{lb}} + \rho_{\mathrm{ub}})$. 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 $\rho_{\mathrm{ub}} = \rho(T)$; otherwise, it updates $\rho_{\mathrm{lb}} = \rho$. Once $\rho_{\mathrm{ub}} - \rho_{\mathrm{lb}}$ falls below σ, BinMDT falls back to a slightly modified version of IncMDT to find the MDT, starting from the best known triangulation with dilation ρ_{ub}. For more details, see also the full version.

3 Empirical evaluation

Now we present experiments to evaluate our algorithms. Code at data are publicly available[1]. 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.

Experiment design 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 [3] The 210 instances (30 for each size $n \in \{10, 20, \ldots, 70\}$) were generated by placing uniformly random points inside a 10×10 square.

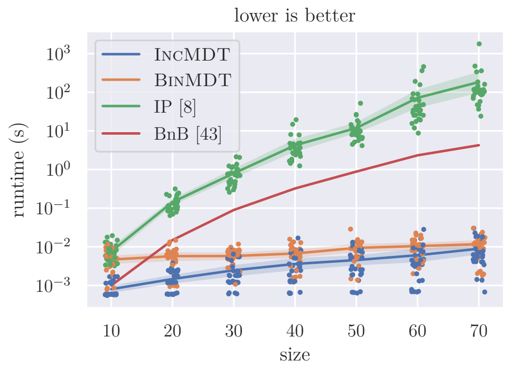
random Two sets of randomly generated instances (total of 800 instances) with points with float coordinates chosen uniformly between 0 and 10^3, ranging from 50 to 10,000 points.

public [7, 6, 20, 19, 10] Well-known publicly available point sets used in the CG:SHOP challenges [7, 6], TSPLIB instances [19], instances from a VLSI dataset [20] and point sets from the Salzburg Database of Polygonal Inputs [10]. In total, we collected 486 instances with up to 10,000 points and an additional 38 with up to 30,000 points.

Comparison to state of the art We compare our approaches to two exact algorithms for the MDT. Note that both use floating-point arithmetic and are not guaranteed to find the optimal solution (although we can confirm that all previous solutions are within a small relative error). The first approach is the IP approach from [3] and the second is the branch & bound (BnB) algorithm from [21].

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 3. All instances are solvable

[1] Code and data: `https://doi.org/10.5281/zenodo.14266122`



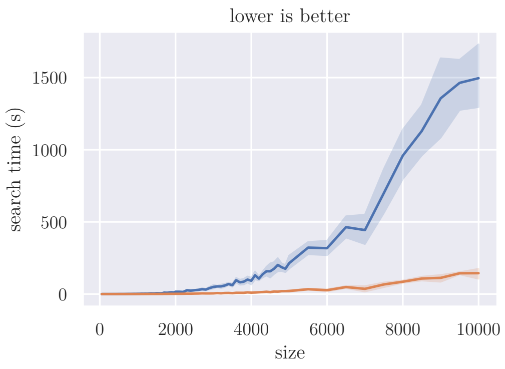
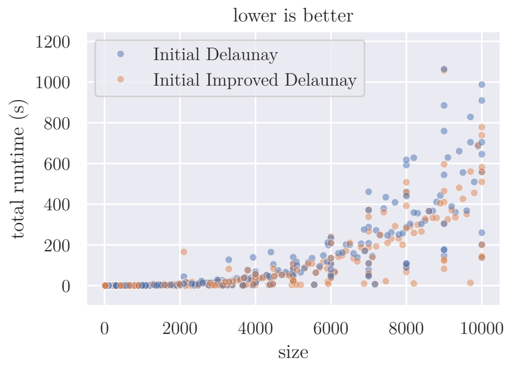


Figure 3 **(Left)** Runtime comparison with the approaches from [3] and [21] on the *random-small* set. **(Right)** BinMDT is significantly faster than IncMDT on the *random* benchmark set.



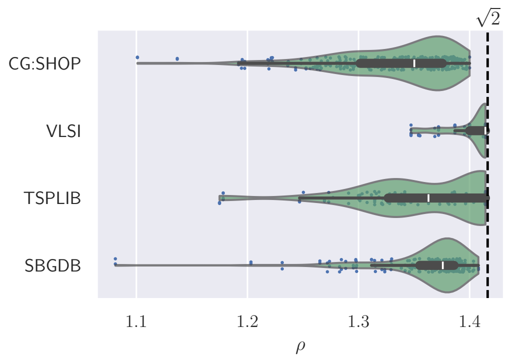


Figure 4 Experiments on the *public* benchmark set. **(Left)** Using the improved Delaunay triangulation as an initial solution significantly improved the performance. **(Right)** The dilation of the MDTs is at most $\sqrt{2}$ for all instances.

in less than 0.1 s. Additionally, [21] provided results for TSPLIB [19] 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.

Algorithm comparison We now compare IncMDT to BinMDT; see the full version for more detail. We conduct our first experiment on the *random* instances with up to 10,000 points; this experiment confirms that BinMDT achieves a significantly lower runtime. For more details, see Figure 3.

We also conduct an additional experiment on the *public* instances up to 10,000 points to determine whether performing greedy, local improvements to the Delaunay triangulation, which we use as initial solution, is worthwhile; see Figure 4. 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.

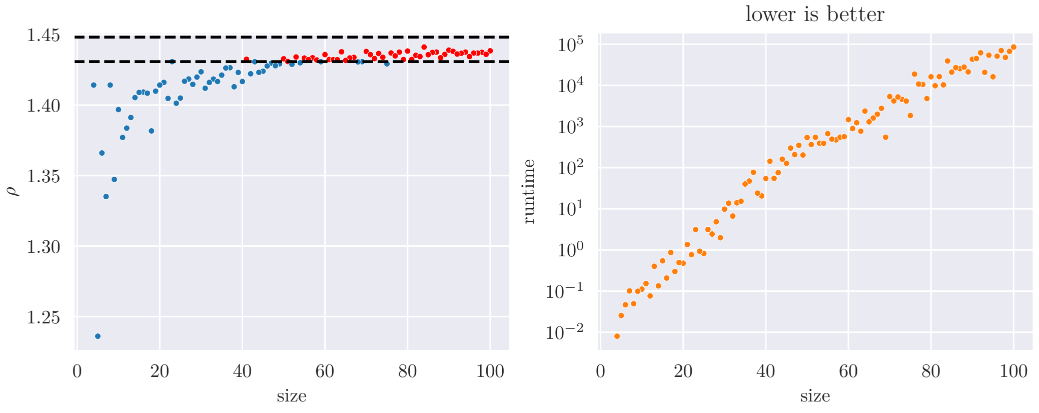


Figure 5 Dilations and runtimes for regular n-gons for $4 \le n \le 100$. Red dots improve the current lower bound of 1.4308 that comes from the regular 23-gon. The dashed black lines mark the known upper bound of 1.4482 and the previous best lower bound of 1.4308.

Regular n-gons The worst-case dilation of a regular n-gon has received considerable attention [17, 9, 22], with a lower bound of 1.4308 and an upper bound of 1.4482. Improving this gap is an open question posed by [9], originating from [2, 13]. With our exact algorithm, we were able to compute bounds for $n \in \{4, 5, \ldots, 100\}$ and found that the dilation of a regular 84-gon is at least 1.44116, see Figures 1 and 5 and the full version for details.

4 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.

References

1 Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean geometry*, pages 47–123. World Scientific, 1995.

2 Prosenjit Bose and Michiel H. M. Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom.*, 46(7):818–830, 2013. URL: https://doi.org/10.1016/j.comgeo.2013.04.002, doi:10.1016/J.COMGEO.2013.04.002.

3 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.

4 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.

5 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications, 3rd Edition*. Springer, 2008. URL: https://www.worldcat.org/oclc/227584184.

6 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 arXiv:2004.04207*, 2020.

7 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.

8 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.

9 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.

10 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.

11 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.

12 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.

13 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.

14 Alexander Klein. Effiziente Berechnung einer dilationsminimalen Triangulierung, 2006.

15 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.

16 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.

17 Wolfgang Mulzer. Minimum dilation triangulations for the regular n-gon. *Master's Thesis. Freie Universität Berlin, Germany.*, 2004.

18 Giri Narasimhan and Michiel H. M. Smid. *Geometric spanner networks.* Cambridge University Press, 2007.

19 G. Reinelt. TSPLIB–A Traveling Salesman Problem Library. *ORSA Journal of Computing*, 3(4):376–384, 1991.

20 A. Rohe. VLSI data set, 2013. URL: https://www.math.uwaterloo.ca/tsp/vlsi/index.html.

21 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.

22 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.

23 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 Evolutinary 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.

24 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`.

25 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`.

26 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/INFCOM.2011.5934879`.