Computing Triangulations with Minimum Stabbing Number

Victor Alvarez * Sándor P. Fekete * Arne Schmidt *

Abstract

For a given point set P or a polygon $\mathcal{P}$, we consider the problem of finding a triangulation T with minimum stabbing number, i.e., a triangulation such that the maximal number of segments hit by any ray going through T is minimized. We prove that this problem is NP-hard; this differs from the problem of triangulating a polygon with minimum edge weight, which is solvable in polynomial time with a simple dynamic program [7]. In an experimental part we test various heuristics.

1 Introduction

Triangulations of point sets or polygons are natural auxiliary structures for a wide range of applications. Depending on the context, a variety of objective functions have been considered to measure their quality. Arising from the context of ray shooting, one such measure that has received a growing amount of attention is the *stabbing number*: This is the maximum number of triangulation edges any line (called a *stab*) can intersect; finding a triangulation of minimum stabbing number corresponds to finding a triangulation that is as "transparent" or "shallow" as possible. This type of question has been considered for a number of structures on a given point set, such as matchings, trees, or triangulations; see Fekete et al. [6].

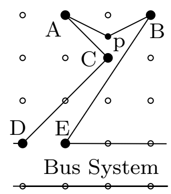
In this paper we consider two variations of stabbing problems: (1) triangulating a point set P or (2) a polygon $\mathcal{P}$ such that the stabbing number is minimized. More formally, the Minimum Stabbing Triangulation Problem (MSTR) asks for a triangulation T of a given point set P, such that the stabbing number $\max_{S\in\mathcal{S}}(|\{e \in T : e \cap S \neq \emptyset\}|)$ is minimal, where $\mathcal{S}$ is the set of all stabs. The problem of triangulating a polygon (MSPT) is defined analogously.

Related Work. Chazelle et al. [3] consider geodesic triangles, i.e., triangles with concave sides. They show that for every polygon P with n vertices, $O(\log n)$ triangles can be hit. Fekete et al. [6] prove NP-hardness of stabbing problems for matchings and trees, and triangulations in [9]. Aichholzer et al. [2] prove NP-hardness of the stabbing problem for polygons. De Berg and van Kreveld [4] study decompositions of rectilinear polygons into rectangles and show that there exists such a decomposition with stabbing number $\mathcal{O}(\log n)$, where the stabbing number counts the number of rectangles any line can intersect. More recently, Piva and de Souza [8] provide a new IP formulation and solve instances with 5000 points [8]. Welzl [11] shows how to construct a spanning tree having a crossing number of $\mathcal{O}(\sqrt{n})$, which is closely related to the stabbing number.

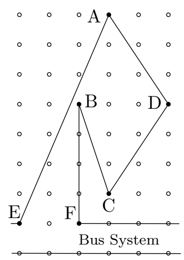
Our Contribution. We prove that the MSPT is NP-hard for axis-parallel stabs; we also present a number of heuristics and experimental results for the MSTR.

2 Triangulating Polygons

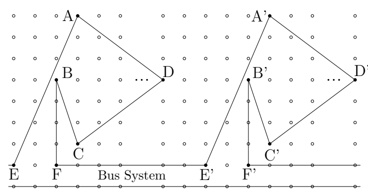
Theorem 1 *The problem MSPT with axis-parallel stabs is* NP*-hard.*



(a) Checker gadget.



(b) Literal gadget.



(c) Variable gadget for variable x with x left and $\overline{x}$ right

Figure 1: Components of the polygon with bus system underneath. Circles represent grid points, filled circles are vertices of the polygon.

Proof. We give a reduction of 3SAT to our problem. Consider a 3SAT instance I consisting of C_I clauses, L_I literals and V_I variables. We transform this instance into a polygon that consists of four components (see Figure 1 for an overview): a *literal gadget* for representing a literal l; a *variable gadget* for representing a variable v; a *checker gadget* for guaranteeing a valid assignment of variables and literals; a *bus system* for connecting all gadgets.

*Department of Computer Science, TU Braunschweig, Germany. {s.fekete,arne.schmidt}@tu-bs.de, alvarez@ibr.cs.tu-bs.de

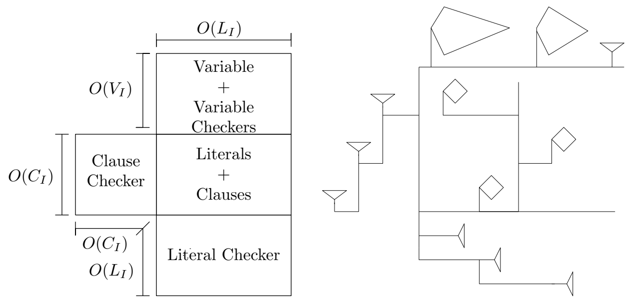


Figure 2: Left: Block model of the 3SAT polygon. Right: Part of a reduction polygon.

The Gadgets. The gadgets are shown in Figure 1. By choosing the diagonals AC or BD we can choose the corresponding literal to be `false` or `true`, respectively. Considering a variable gadget the setting will be the other way round: Choosing AC and $B'D'$ sets the variable to `true` and choosing $A'C'$ and BD sets the variable to `false`. The checker gadgets force a correct setting of all diagonals, e.g., a variable x cannot be `true` and `false` at the same time. By scaling the grid we are allowed to insert more points in the checker gadget and thus, increase the maximum stabbing number.

The Construction. We omit full details due to space constraints. Consider the block model in Figure 2. We place the gadgets corresponding to the blocks; the clauses are represented by literals in the middle, variables are placed on top, literal checkers below and clause checkers to the left. Figure 2 shows part of an example.

Stabbing Number. For all stabs the stabbing number is at most $4C_I$ (proof omitted due to space limitations). Thus, if the 3SAT instance is satisfiable then the stabbing number of the triangulation of the polygon is at most $4C_I$.

We can also show the other direction: if there is a triangulation with stabbing number $\leq 4C_I$, then the 3SAT instance is satisfiable. Finally, it is straightforward to check that the overall construction has polynomial size. □

3 Integer Programm Formulation

When trying to solve the problem with an IP, it is natural to focus on the maximal number of non-crossing edges, but using triangles is the better choice [10]. This IP may have $\Omega(n^3)$ variables, however, we only need to consider empty triangles that do not enclose any point. The basic idea of the IP is that each edge e in a triangulation is part of two triangles if e is not part of the convex hull. If e is a part of the convex hull then there is exactly one triangle having e as an edge.

Definition 1 *We define the edges of the convex hull $E_H := \{e \in E | e \text{ is on the convex hull}\}$. $\Delta(P)$ is the set of all empty triangles induced by point set P. $\delta^-(ij) := \{ijl \in \Delta(P) | ijl \text{ is a right turn}\}$ and $\delta^+(ij) := \{ijl \in \Delta(P) | ijl \text{ is a left turn}\}$. Moreover, for a triangle ijl and a stab S, let $c^S_{ijl} := \beta^S_{ij} + \beta^S_{jl} + \beta^S_{il}$, where*

$$\beta^S_{ij} := \begin{cases} 1, & \text{if } ij \in E_H \text{ and } ij \text{ intersects } S \\ 0.5, & \text{if } ij \notin E_H \text{ and } ij \text{ intersects } S \\ 0, & \text{else} \end{cases}$$

With this, we can formulate a triangle based IP for MSTR (analogously for MSPT) as follows:

min K

s.t.:

$$\forall ij \in E \setminus E_H : \sum_{\substack{ijl \in \Delta(P) \\ ijl \in \delta^+(ij)}} x_{ijl} = \sum_{\substack{ijl \in \Delta(P) \\ ijl \in \delta^-(ij)}} x_{ij}$$

$$\forall ij \in E_H : \sum_{ijl \in \Delta(P)} x_{ijl} = 1$$

$$\forall S \in \mathcal{S} : \sum_{\substack{ijl \in \Delta(P): \\ ijl \cap S \neq \emptyset}} c^S_{ijl} x_{ijl} \leq K$$

$$\forall t \in \Delta(P) : x_t \in \mathbb{B}$$

4 Heuristics

In addition to exact methods, we also tested a number of heuristics: the new *BER*-algorithm (Section 4.1), local optimization based on *flipping* (Section 4.2), and an algorithm for obtaining a smaller edge set (Section 4.3).

4.1 Bridge Error Rate (BER)

The idea of the *Bridge Error Rate* (BER) algorithm is as follows. For each stab S there is a lower bound on the number of edges crossing S (Theorem 3). If we insert an edge e then we may raise this lower bound. Because we do not want to increase the lower bound, we greedily add edges that raise it as little as possible.

Definition 2 *(Bridge) Consider a stab S, as shown in Figure 3. S splits the point set P into subsets P_M, P_U and P_L, i.e., points on S and in the upper and lower half-space, respectively. For a connected subset $\mathtt{CH}^*(P_U) \subseteq \mathtt{CH}(P_U)$, where $\mathtt{CH}(V)$ is the convex hull of a set V, a point p is in $\mathtt{CH}^*(P_U)$ iff there is a point $q \in P \setminus P_U$ such that $\{p, q\}$ does not cross $\mathtt{CH}(P_U)$. Analogously, we define $\mathtt{CH}^*(P_L)$. The bridge $B(S)$ is the union of $\mathtt{CH}^*(P_U)$, $\mathtt{CH}^*(P_L)$ and P_M. Furthermore, we define $|B(S)|$ as the number of points in $B(S)$. Note that $\mathtt{CH}^*$ may not be an upper or lower envelope of the convex hull.*

Theorem 3 *Consider a stab S and its bridge $B(S)$. Let m be the number of points on S, and m^* the*

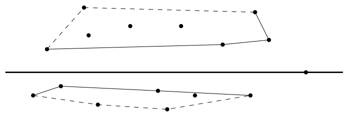


Figure 3: A stab (solid fat line). The solid thin line is $\mathtt{CH}^*(P_L)$ and $\mathtt{CH}^*(P_R)$, respectively. The dashed lines are the extensions to the convex hull on each side.

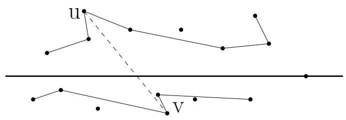


Figure 4: A constrained bridge. The end points of $\{u, v\}$ are connected to $B(S)$ via geodesic paths.

number of points on S that are not part of $\mathtt{CH}(P)$. Then there are at least $|B(S)| + m + m^* - 1$ edges intersecting S.

If an arbitrary edge e is added that intersects a stab S and some edges in $B(S)$, we remove these edges from $B(S)$ and add geodesic paths connecting the end points of e and $B(S)$ (see Figure 4). This results in a new *constrained* bridge, yielding a new lower bound. Note that Theorem 3 also holds for the constrained bridge. Having constrained bridges, we can insert one edge after another until we have a triangulation. Further details are omitted due to limited space.

4.2 Flipping

Flipping an edge in a triangulation is a natural heuristic approach to local optimization. We flip edges only when the flip does not increase the stabbing number of any stab, and there is a stab with maximum stabbing number whose number decreases after the flip. If no such edge exists, we flip edges as long as the stabbing number of a stab is not increased and the total edge length decreases. As described in Section 5, with these criteria we achieve near-optimal solutions.

4.3 Smaller Edge Sets

When solving the IP for our problems, the runtime to solve an instance depends on the number of variables, i.e., the number of edges. An idea for using a reduced edge set comes from the IP for triangulations. We begin with the convex hull and add edges from the smallest triangle in terms of edge length. From these new edges we repeat the procedure in the left and right half-space from each supporting line. A reduced edge set can be seen in Figure 5.

5 Experiments

Our experiments were run on 64-bit Ubuntu 14.04.4 LTS with an Intel Core i7-4770 @3.4GHz with 32KB

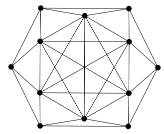


Figure 5: Example for an edge set with 12 points. From 66 possible edges we now only use 33.

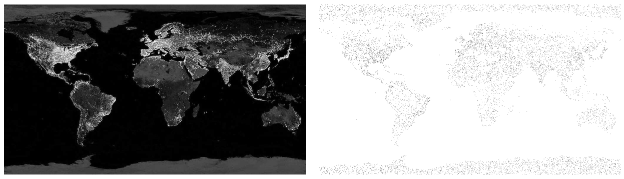


Figure 6: Left: Satellite image of earth by night [1]. Right: Generated image with 10000 points.

solved	# points
30	170
15	190
<15	>190

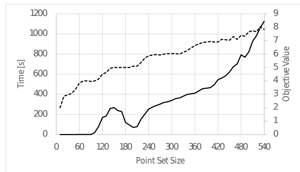


Figure 7: Left: Number of solved instances (out of 30) for different point sizes. Right: Average time and objective value of solvable instances. The dashed line represents the objective value, the solid line the average time in seconds.

L1 cache, 256KB L2 cache, 8192KB L3 cache and 32GB RAM. We used the g++ compiler version 4.8.4 and optimization flag -O3.

Clustered Instances. Clustered instances were generated from a lightmap, similar to the description in Fekete et al. [5]. For a given illumination map, the brightness value induce a density function that can be used for random sampling of points from the image. For the image shown in Figure 6, we created 30 instances per point set size for obtaining *clustered* instances. We omit other instance types.

Optimal Solutions. The left of Figure 7 shows how many instances (out of 30) with n points are solvable within the 30-minute time limit. The graph on the right side shows average runtime and average objective value for each number of points.

Triangulation Heuristics. Figure 8 shows a comparison of the optimum and the BER-algorithm with flipping. The solutions are quite close to the optimum; the maximum difference is 3. We also tested other methods (e.g., Delaunay triangulations) which resulted in slightly worse solutions.

Reduced Edge Sets. Consider Figure 9. We observe that the reduced edge set appears to grow (close to)

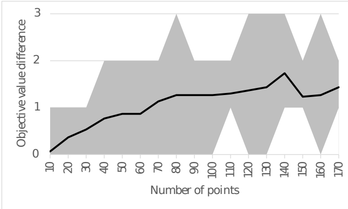


Figure 8: Difference between the optimal objective value and the result of the BER algorithm with flipping. Shaded area shows the range. The solid line shows average difference.

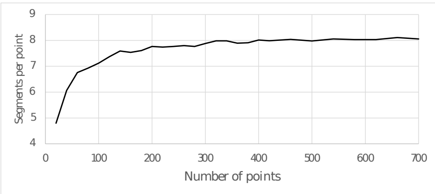


Figure 9: Number of edges per point in the reduced edge set on clustered instances.

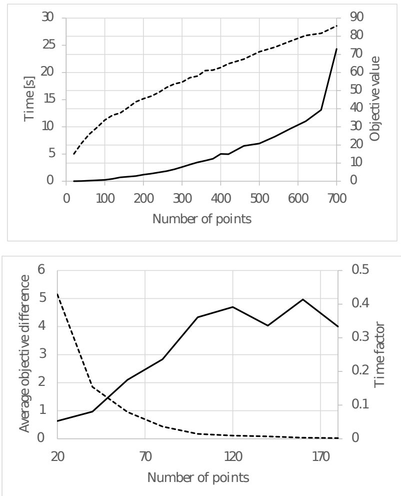


Figure 10: Top: Average time (solid line) and stabbing number (dashed line) using a reduced edge set. Bottom: Difference of stabbing number of an optimal solution and with reduced edge set (solid line) and the time ratio reduced edge set / optimum.

linearly in the number of points. Figure 10 shows a comparison between optimum and the solutions with reduced edge set. The speedup is outstanding; for example, the computing time is reduced by a factor of 500 for 180 points. Note that this comes at the expense getting a suboptimal solution.

6 Future Work

We have shown that the problem MSPT of finding a triangulation of a polygon with minimum stabbing number is NP-hard. Our experiments for the MSTR show that flipping edges can yield excellent solutions. This is also the case for our BER-algorithm; however, its runtime is rather high. We can also use a reduced edge set, resulting in a massive speedup with an only slightly increased stabbing number.

An interesting challenge on the theoretical side lies in developing approximation algorithms, if there are any; this is already an open problem for the other problems of stabbing type considered in [6]. On the experimental side, it is also of interest to collect hard instances, i.e., instances that are practically difficult to solve to optimality.

References

[1] Blick.ch: Lichter der Erde. `http://www.blick.ch/life/wissen/nasa-bilder-aus-dem-all-lichter-der-erde-id2131506.html`. Accessed: 2016-06-21.

[2] O. Aichholzer, F. Aurenhammer, T. Hackl, F. Hurtado, A. Pilz, P. Ramos, J. Urrutia, P. Valtr, and B. Vogtenhuber. On k-convex point sets. *Computational Geometry*, 47(8):809–832, 2014.

[3] B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.

[4] J. A. de Loera, S. Hosten, F. Santos, and B. Sturmfels. The polytope of all triangulations of a point configuration. *Documenta Mathematica*, 1(4):103–119, 1996.

[5] S. P. Fekete, A. Haas, M. Hemmer, M. Hoffmann, I. Kostitsyna, D. Krupke, F. Maurer, J. S. B. Mitchell, A. Schmidt, C. Schmidt, and J. Troegel. Computing nonsimple polygons of minimum perimeter. In *15th International Symposium on Experimental Algorithms (SEA 2016)*, pages 134–149, 2016.

[6] S. P. Fekete, M. E. Lübbecke, and H. Meijer. Minimizing the stabbing number of matchings, trees, and triangulations. *Discrete & Computational Geometry*, 40(4):595–621, 2008.

[7] G. Klincsek. Minimal triangulations of polygonal domains. *Ann. Discrete Math*, 9:121–123, 1980.

[8] B. Piva and C. C. de Souza. Minimum stabbing rectangular partitions of rectilinear polygons. *Computers & Operations Research*, 80:184–197, 2017.

[9] B. Piva, S. P. Fekete, and C. de Souza. On triangulations with minimum stabbing or minimum crossing number. Manuscript, 2016.

[10] A. Tajima. Optimality and integer programming formulations of triangulations in general dimension. In *International Symposium on Algorithms and Computation*, pages 378–387. Springer, 1998.

[11] E. Welzl. On spanning trees with low crossing numbers. In *Data structures and efficient algorithms*, pages 233–249. Springer, 1992.