Computing Crossing-Free Configurations with Minimum Bottleneck*

Sándor P. Fekete[1] and Phillip Keldenich[1]

1 **Department of Computer Science, TU Braunschweig, Germany**
{s.fekete,p.keldenich}@tu-bs.de

Abstract

We consider problems of finding non-crossing *bottleneck* structures for a given planar point set: For a given a set of vertices V, the problem Minimum Bottleneck Polygon (MBP) is to find a simple polygon P with vertex set V whose longest edge is as short as possible; the problem Minimum Bottleneck Simple Matching (MBSM) is to find a crossing-free matching of V whose longest edge is as short as possible. Both problems are known to be NP-complete and neither admits a PTAS. We develop exact methods that can solve benchmark instances (newly generated and from the classic TSPLIB library) with up to 1,500 points for MBP and up to 20,000 points for MBSM to provable optimality.

1 Introduction

Finding a simple polygon with a given set V of vertices in the plane is one of the basic problems of computational geometry. If we want to minimize the overall length, this is is equivalent to the classic Traveling Salesman Problem (TSP), as a shortest tour is always non-crossing. However, if the objective is to minimize the length of the longest edge, this is no longer the case, see Fig. 1. This problem Minimum Bottleneck Polygon (MBP) is NP-complete and unless P=NP, it cannot be approximated within a factor better than $\sqrt{3}$, as it is NP-complete to decide whether a hexagonal grid graph has a Hamiltonian cycle (HC) of unit edges (see Arkin et al. [5]). We are not aware of any constant-factor approximation algorithms for the MBP.

A similarly basic geometric optimization problem is to find a matching for a given vertex set. When minimizing the total length of all edges, an optimal solution must also be non-crossing; this allows it to use standard matching techniques, subject to the (purely theoretical) issue of computing the sum of a set of square roots. Matching techniques can also be used to compute a Minimum Bottleneck Matching (MBM) in polynomial time. However, a solution to MBM does not have to be non-crossing, as shown in Fig. 1. In fact, it was shown by Abu-Affash et al. [2] that this problem Minimum Bottleneck Simple Matching (MBSM) is NP-complete and does not allow a PTAS. They also provide a $2\sqrt{10} \approx 6.325$-approximation algorithm and state without proof that they can reduce this factor to $(1+\sqrt{2})\sqrt{5} \approx 5.398$.

In this paper, we develop methods for computing provably optimal solutions for benchmark instances up to 2,000 points. Beyond illustrating the practical solvability of both problems, this will provide ground truth for testing potential (improved) approximation methods.

Related work. There is a huge body of related work; due to limited space, we only mention a small subset.

* This work was partially supported by the DFG Research Unit "Controlling Concurrent Change", funding number FOR 1800, project FE407/17-2, "Conflict Resolution and Optimization".

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.
This is an extended abstract of a presentation given at EuroCG'18. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

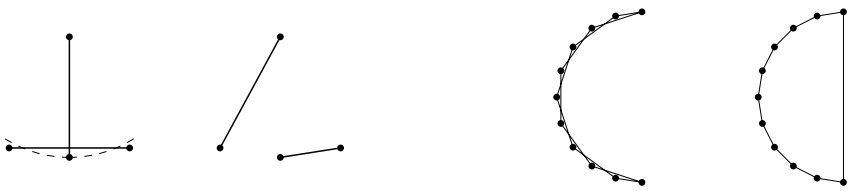

Figure 1 Left: A minimum bottleneck matching and a crossing-free minimum bottleneck matching. Right: A minimum bottleneck tour and a minimum bottleneck polygon.

The minimum bottleneck TSP was first introduced by Gilmore and Gomory [9]. For metric instances, there is a 2-approximation algorithm implied by Fleischner's theorem [8, 12] which states that the square of every two-connected graph is Hamiltonian; for general metric instances, this factor is best possible. Hochbaum and Shmoys [10] also prove a factor of 2 within a framework providing approximation algorithms for several bottleneck problems. The problem of finding a *longest* simple polygon for a given vertex set was considered by Alon et al. [3]; they conjecture this problem to be NP-hard, but this is still open. See Dumitrescu and Tóth [6] for improved approximation factors.

2 Minimum Bottleneck Polygonalization

2.1 Modeling

We start with a basic formulation NMBP(V) of Minimum Bottleneck Polygon as a Mixed Integer Program, where x_{pq} is a Boolean variable encoding whether pq is an edge of the polygon and B encodes the bottleneck of the solution. For two points $p, q \in V$, let $\chi(pq)$ be the set of line segments crossing pq.

$$\min B \text{ s. t.}$$

$$\forall p \in V : \sum_{q \neq p} x_{pq} = 2 \tag{1}$$

$$\forall p, q \in V, rs \in \chi(pq) : x_{pq} + x_{rs} \leq 1 \tag{2}$$

$$\forall \emptyset \subsetneq S \subsetneq V : \sum_{p \in S, q \in V \setminus S} x_{pq} \geq 2 \tag{3}$$

$$\forall p, q \in V : \|pq\|_2 \cdot x_{pq} \leq B \tag{4}$$

$$x_{pq} \in \{0, 1\}$$

Degree constraints (1) ensure two incident edges for each point, while *crossing constraints* (2) exclude crossing edges. The *subtour constraints* (3) enforce a connected solution. Finally, the *bottleneck constraints* (4) enforce the maximum edge length B.

This naïve formulation is only practical for small point sets. Firstly, it is well known that using an auxiliary variable B to encode a min max-type objective often induces weak LP relaxations and thus leads to a suboptimally large search tree. Moreover, the total number of crossing constraints corresponds to the number of convex quadruples in V, which is known to be $\Omega(n^4)$ (see [11, 14]), so using all these constraints at once becomes prohibitively expensive.

In the following, we present a formulation of the MBP as a sequence BMBP$_\tau(V)$ of IPs addressing these issues; this resembles the approach used in [7] for determining the threshold value for a triangulation whose shortest edge is as long as possible. For a threshold value τ, BMBP$_\tau(V)$ is integer feasible iff V has a polygon with bottleneck at most τ; we exclude all edges longer than τ. As before, we use degree constraints (5), subtour constraints (7) and

crossing constraints (6). Thus, a minimum bottleneck polygon can be found with binary search over possible values of τ. The optimal bottleneck is the length of an edge; therefore, this is a discrete set of possible values.

$$\min \sum_{p,q \in V, \|pq\|_2 \leq \tau} \|pq\|_2 \cdot x_{pq} \text{ s. t.}$$

$$\forall p \in V : \sum_{q \in V, \|pq\|_2 \leq \tau} x_{pq} = 2 \tag{5}$$

$$\forall p, q \in V, \|pq\|_2 \leq \tau, rs \in \chi(pq) : x_{pq} + x_{rs} \leq 1 \tag{6}$$

$$\forall \emptyset \subsetneq S \subsetneq V : \sum_{p \in S, q \notin S, \|pq\|_2 \leq \tau} x_{pq} \geq 2 \tag{7}$$

$$x_{pq} \in \{0, 1\}$$

In our implementation of this formulation, only violated crossing and subtour constraints are added iteratively by generating appropriate cutting planes. This results in only small subsets of these large families actually being used. This is aided by our use of the objective function. Instead of directly minimizing the bottleneck, we minimize the sum of all edge lengths. Due to the triangle inequality, most avoidable edge crossings never occur in intermediate (fractional and integral) solutions. Moreover, we do not have to solve $\text{BMBP}_\tau(V)$ to optimality; we can abort the search as soon as the first integer feasible solution is found.

2.2 Computational Results

We implemented $\text{NMBP}(V)$ and $\text{BMBP}_\tau(V)$ in C++, using IBM ILOG CPLEX 12.6.2 as our IP solver. All our experiments ran on a workstation running Linux 4.4 on an Intel Core i7-6700K CPU at 4 GHz clock frequency with 64 GiB of RAM. In order to efficiently construct $\text{BMBP}_\tau(V)$, we used an implementation of *kd*-trees provided by CGAL [1] to enumerate all points within distance τ of a query point. Moreover, violated crossing constraints are detected using CGAL's sweep line implementation. To compare the performance of NMBP and BMBP, we ran both NMBP and BMBP on a set of small instances. Using NMBP, most instances with more than 50 points cannot be solved within 500 seconds, while BMBP solves these instances in less than half a second. Thus, we only evaluate BMBP in the remainder of the section. Fig. 2 shows the results of running BMBP on randomly generated point sets with a modest time limit of 10 minutes. We also ran BMBP on all geometric instances from the classic TSPLIB [13] with fewer than 2,500 points. Within a time limit of one hour, we were able to solve most of them to optimality; see Fig. 3.

3 Minimum Bottleneck Matching

3.1 Modeling

By modifying the right-hand side of the degree constraints (1) and (5) to 1 and removing the subtour constraints (3) and (7), we obtain a formulation of the crossing-free minimum bottleneck matching problem as naïve MILP $\text{NMBM}(V)$ and as sequence of IPs $\text{BMBM}_\tau(V)$. In order to improve its performance, we generate blossom constraints

$$\forall S \subsetneq V, |S| \text{ odd}: \sum_{p \in S, q \notin S, \|pq\|_2 \leq \tau} x_{pq} \geq 1, \tag{8}$$

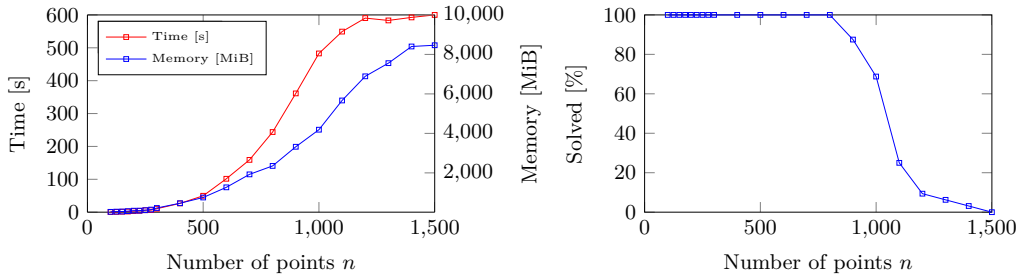

Figure 2 Left: Average running time and peak memory usage for BMBP, run with a time limit of 600 s on point sets generated uniformly at random. Right: Percentage of instances solved within the time limit.

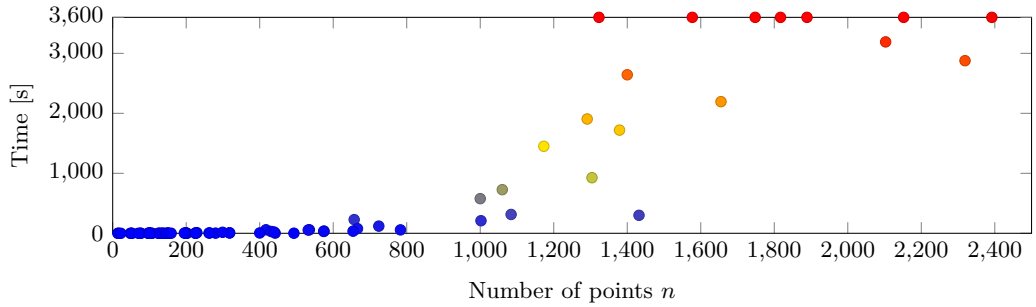

Figure 3 Time required by BMBP to solve the TSPLIB instances; computation was aborted after one hour.

as cutting planes. We identify violated blossom inequalities by searching for odd components in the support graph of a fractional solution. In order to further restrict the search space for the binary search, we implemented a minimum bottleneck matching algorithm to serve as a lower bound and a crossing removal heuristic to produce an initial crossing-free solution as an upper bound; see Section 3.2.

3.2 Crossing Repair Heuristic

A straightforward way to heuristically turn a crossing matching into a non-crossing one is to use a sequence of local 2-OPT exchanges, replacing a crossing pair of edges pq, rs by pr, sq or ps, qr. Any 2-OPT step decreases the sum of edge lengths, so this process must terminate with a non-crossing matching. However, this heuristic does not seem to perform well with respect to the bottleneck.

An alternative is a simple but effective heuristic for converting a crossing matching M_C with bottleneck B into a non-crossing matching M_{NC}, while trying to keep the bottleneck edge as short as possible. We use a standard sweep line algorithm to detect crossings. If there is no more crossing, we are done. Otherwise, we pick an arbitrary crossing pq, rs. Using a *kd*-tree, we perform a simultaneous incremental nearest-neighbor search, starting from p, q, r and s, constructing a set of close points N that contains up to K points, for some constant K; we use $K = 50$ in our experiments. Whenever a new point is discovered, it is added to N, together with its matching partner in M_C. Once N contains K points, we compute the bounding box of N and extend it by B in every direction. We use a range query

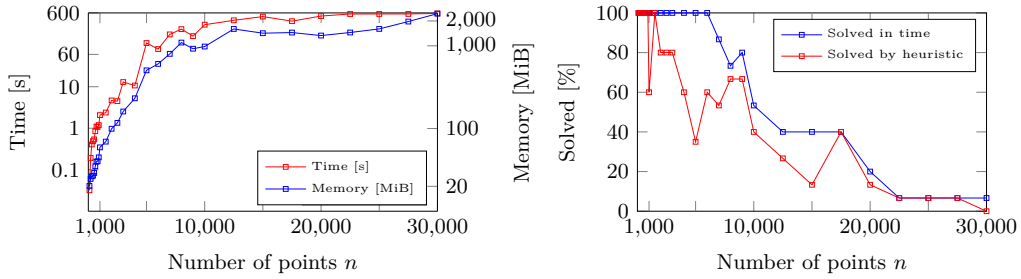

Figure 4 Left: Average running time and peak memory usage for BMBM, run with a time limit of 600 s on point sets generated uniformly at random. Right: Percentage of instances solved within the time limit, and percentage of instances solved to optimality using the crossing-removal heuristic.

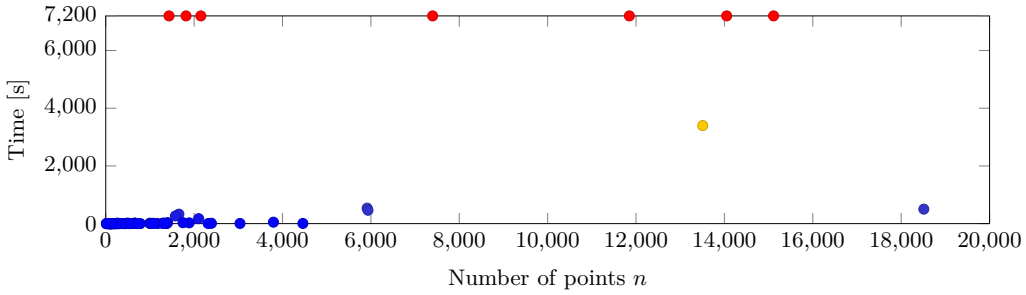

Figure 5 Time required by BMBM to solve the TSBLIB instances; the time limit was two hours.

to find all points inside the extended bounding box; this set of points consists of our *internal* points N and *external* points T. For all external points, we find the corresponding matching edge in M_C; this gives us a set of edges E_T. We use $\text{BMBM}_\tau(N)$ with binary search on τ to find a minimum bottleneck crossing-free matching on N; however, in order to avoid introducing new crossings, we prohibit using any edge that crosses an edge of E_T, unless this edge is part of M_C. In M_C, we replace the matching edges corresponding to points in N with the edges from the resulting matching. In this way, the crossing pq, rs disappears and no new crossings can appear. We iterate this procedure until there are no more crossings; the number of crossings is reduced by at least one in each iteration, thus the heuristic terminates with a crossing-free matching M_{NC}. In certain situations, this heuristic can fail, because a crossing-free matching cannot be found due to the forbidden edges. In this case, we resort to performing 2-OPT steps to remove some crossings before continuing to use the original heuristic.

3.3 Computational Results

We implemented both NMBM and BMBM and evaluated them under the same circumstances as outlined in Section 2.2. Similar to the situation for polygons, the naïve NMBM cannot compete with BMBM, so we give only give computational results BMBM. We were able to solve almost all TSPLIB instances with up to 20,000 points within a time limit of two hours (see Fig. 5). For point sets chosen uniformly at random from the unit square, we were able to solve all generated instances with up to 6,000 points and most instances with up to 10,000 points within ten minutes (see Fig. 4).

In many instances, applying our crossing removal heuristic to a minimum bottleneck matching yields a crossing-free solution with the same bottleneck (see Fig. 4), thus resulting in a provably optimal solution. For all randomly generated instances, the minimum bottleneck

was achievable in a crossing-free manner; on these instances, our crossing repair heuristic was off by a factor of at most 2.215 (this factor was 1.11 on average with median 1.024).

4 Future Work

We presented exact approaches for both the MBP and the MBSM. Many interesting theoretical and practical problems remain that are left for future work.

The most interesting theoretical problem is to develop a constant-factor approximation algorithm for MBP. In our experiments, we found that for large point sets generated uniformly at random, a minimum bottleneck matching can always be achieved with a non-crossing solution. Is there an analytic basis for this observation? For general point sets, it may be interesting to explore the properties of the matching polytope with added crossing constraints.

On the practical side, the quadratic number of edge variables is the biggest impediment for solving larger instances. For the classic TSP, this has been dealt with by using column generation and related methods [4]. For MBP, there are potentially many additional constraints that could be used for cutting plane generation. Doing this in an efficient manner requires rewriting large parts of the integer programming solver. For MBSM, more sophisticated algorithms may be able to identify more (helpful) blossom constraints.

References

1 The Computational Geometry Algorithms Library. `http://www.cgal.org`.

2 A. K. Abu-Affash, P. Carmi, M. J. Katz, and Y. Trabelsi. Bottleneck non-crossing matching in the plane. *Computational Geometry*, 47(3):447–457, 2014.

3 N. Alon, S. Rajagopalan, and S. Suri. Long non-crossing configurations in the plane. *Fundam. Inform.*, 22(4):385–394, 1995.

4 D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A computational study.* Princeton university press, 2011.

5 E. M. Arkin, S. P. Fekete, K. Islam, H. Meijer, J. S. Mitchell, Y. Núñez-Rodríguez, V. Polishchuk, D. Rappaport, and H. Xiao. Not being (super)thin or solid is hard: A study of grid Hamiltonicity. *Computational Geometry*, 42(6–7):582–605, 2009.

6 A. Dumitrescu and C. D. Tóth. Long non-crossing configurations in the plane. *Discrete & Computational Geometry*, 44(4):727–752, 2010.

7 S. P. Fekete, W. Hellmann, M. Hemmer, A. Schmidt, and J. Troegel. Computing MaxMin edge length triangulations. In *Proc. 17th Worksh. Alg. Eng. Exp. (ALENEX)*, pages 55–69, 2015. Full version to appear in *Journal of Computational Geometry*.

8 H. Fleischner. The square of every two-connected graph is Hamiltonian. *Journal of Combinatorial Theory, Series B*, 16(1):29–34, 1974.

9 P. C. Gilmore and R. E. Gomory. Sequencing a one state-variable machine: A solvable case of the Traveling Salesman Problem. *Operations Research*, 12(5):655–679, 1964.

10 D. S. Hochbaum and D. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33:533–550, 07 1986.

11 L. Lovász, K. Vesztergombi, U. Wagner, and E. Welzl. Convex quadrilaterals and k-sets. In *Contemporary Mathematics Series, 342, AMS 2004*, pages 139–148, 2004.

12 R. G. Parker and R. L. Rardin. Guaranteed performance heuristics for the bottleneck Travelling Salesman Problem. *Operations Research Letters*, 2(6):269–272, 1984.

13 G. Reinelt. TSPlib — A Traveling Salesman Problem library. *ORSA J. Computing*, 3(4), 1991.

14 E. R. Scheinerman and H. S. Wilf. The rectilinear crossing number of a complete graph and Sylvester's "four point problem" of geometric probability. *The American Mathematical Monthly*, 101(10):939–943, 1994.