Covering Geometric Sets with Lines*

Sándor P. Fekete[1], Chek-Manh Loi[1], and Michael Perk[1]

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

Abstract

We consider the Set Cover Problem for geometric neighborhoods: Given a family $\mathcal{R} = \{R_1, \ldots, R_n\}$ of n connected regions in the plane, find as few lines as possible, such that each region is intersected by some line. Even special cases of this problem are known to be NP-complete, and a spectrum of work has focused on theoretical results such as approximation algorithms; previous practical work has been limited, and included the case in which each R_i is a single point, and the task is to decide whether a small number of lines suffice. We present exact methods for a variety of more general scenarios, including provably optimal solutions for sets with up to 2000 points, and near-optimal solutions for sets of polygons with up to 650 polygonal regions and a total of about 4000 vertices.

1 Introduction

The Set Cover Problem (SCP) is an NP-complete problem of fundamental importance, both in theory and practice. For general instances, the greedy algorithm [7] provides an $\mathcal{O}(\log n)$-approximation algorithm, which is best possible in the worst case, unless P=NP. Many variants of the SCP are geometric, e.g., using line segments, rays, convex polygons or star-shaped polygons for covering point sets, lines or other geometric objects. The geometry of an SCP may be helpful: For covering discrete point sets by lines with a limited number of directions we can get constant-factor approximation algorithms [13]. The underlying geometry can also give rise to additional difficulties: As shown by Abrahamsen et al. [1], the Art Gallery Problem (which amounts to covering a simple polygon by a minimum number of star-shaped subpolygons) is $\exists\mathbb{R}$-complete, making it unlikely that it even belongs to NP.

Our Contributions

We study *practically* useful methods for covering a set $\mathcal{R}$ of n geometric regions by a smallest number of lines, see Figure 1 for examples. In particular, we provide the following results.

- Different methods for efficiently computing a discrete set of candidate lines that limit the size of the ensuing set cover instance.
- Exact approaches for near-optimal solutions for covering points or polygons with lines.
- An experimental evaluation for a wide spectrum of benchmark instances.

Related Work

There is a large body of work on geometric Set Cover and Hitting Set problems, so we only point to a very limited selection of most closely related work; for a more extensive overview, see the relatively recent paper by Fekete et al. [13], who considered covering a finite set of points in the plane by a minimum number of lines with a limited number of different slopes.

* This work was supported by DFG project "Computational Geometry: Solving Hard Optimization Problems" (CG:SHOP), FE407/21-1.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. 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.

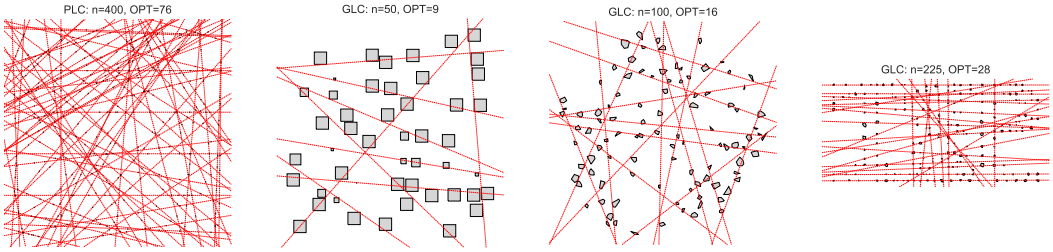

Figure 1 Small example instances that were solved optimally. From left to right: A PLC instance with 400 points and an optimal solution with 76 lines. Three GLC instances (randomly generated squares on a grid, randomly generated polygons in a circle, and the TSPLIB tsp225 [5] instance) and their optimal solutions.

The Point Line Cover (PLC) problem [22] asks for a smallest set of lines to cover a given set of points. It was shown to be NP-hard [25], APX-hard [6] and Max-SNP Hard [23]. Grantson and Levcopoulos [18] developed an $\mathcal{O}(\log OPT)$ algorithm for the PLC. Hassin and Megiddo [19] studied hitting geometric objects with the fewest lines having a small number of distinct slopes. Gaur and Bhattacharya [15] considered covering points with axis-parallel lines in d dimensions. Many other problems related to finding a small set of lines that hit a given set of objects have also been studied; see, e.g., [8, 9, 12, 16, 17, 21, 24].

When considering the coverage of geometric regions (i.e., neighborhoods) instead of discrete points, Aronov et al. [3] provide an $\mathcal{O}(\log \log OPT)$-approximation for hitting set for axis-aligned rectangles and axis-aligned boxes in 3D, based on ϵ-nets. Hitting a set of unit disks has been considered for finding a minimum number of relays for connecting a given set of relays [10]. While the simple greedy approximation algorithm is efficient and worst-case optimal, a logarithmic approximation factor is not good enough in practice. Estivill-Castro et al. [11] evaluated implementations for the PLC, but only considered cases in which the optimal solution is known to be small, i.e., $OPT \leq 7$.

2 Preliminaries

Given a point set $\mathcal{P} \subset \mathbb{R}^2$ of size n, the Point Line Cover Problem (PLC) asks for a smallest set of lines that covers all points in $\mathcal{P}$. We denote the set of all possible lines as $\mathcal{L}$. In the General Line Cover Problem (GLC), we are given a set of disjunct regions $\mathcal{R}$ and ask for a smallest set of lines that intersect all regions in at least one point. It is easy to see that a line intersects a region iff it intersects its convex hull, so we can restrict ourselves to *convex* regions; moreover, we focus on *compact, disjoint* regions with $\mathcal{O}(1)$ complexity.

3 Computing a Candidate Set $\mathcal{L}$

Computing optimal solutions can be subdivided into two steps, as follows.

1. Compute a discrete set $\mathcal{L}$ of candidate lines that contains an optimal line cover.
2. Solve the resulting Set Cover instance.

In theory, the first step is comparatively easy, while the second is NP-hard. However, the focus on practical computation implies that merely polynomial-time computation of a candidate set is insufficient, especially when aiming for small input for the ensuing SCP. A variety of possible approaches for solving those SCP instances is considered in Section 4.

3.1 Point Line Cover

For the PLC, Gajentaan and Overmars [14] showed that determining whether a set of n points in the plane has three collinear points is 3SUM-hard. Even though subquadratic algorithms for 3SUM exist [2, 4], the improvements over $\mathcal{O}(n^2)$ are often times marginal. We propose an $\mathcal{O}(n^2)$ algorithm for determining $\mathcal{L}$ for a given set of points $\mathcal{P}$. The algorithm iterates over all combinations of two points $p_i, p_j \in \mathcal{P}$ and calculates a tuple $t_{ij} = (m, b)$ with m being the gradient of the line ℓ between p_i, p_j and b being the y value of ℓ at $x = 0$. While iterating, we calculate a hash $h(t_{ij})$ to identify every line in $\mathcal{L}$. This allows us to add all collinear points to a set within a single pass over all point pairs.

3.2 General Line Cover

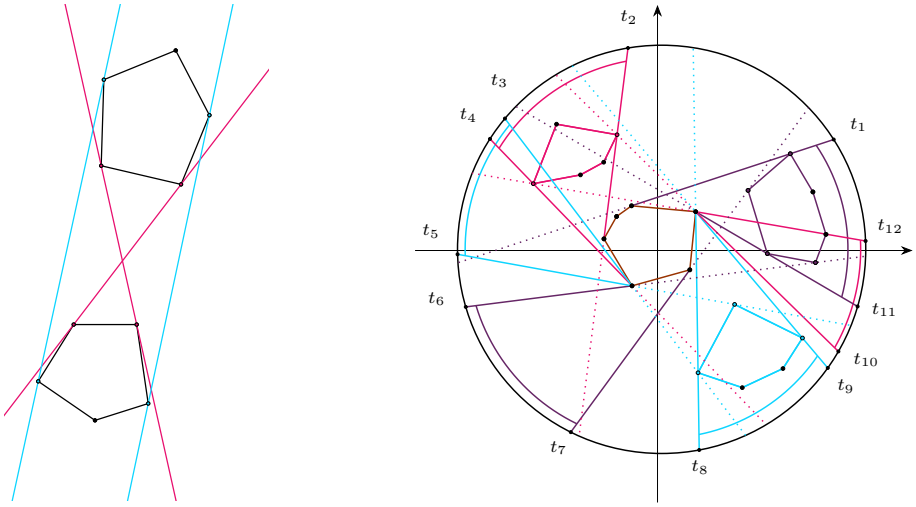

Figure 2 (Left) Four tangents from Lemma 3.1. The red tangents are interior, the blue ones exterior. In this example each polygon has four tangential points. (Right) Rotating tangent algorithm to compute $\mathcal{L}$.

Tangent Lines. For two disjoint compact, convex regions in the GLC, we can reduce the set of candidate lines to two exterior tangents using Lemma 3.1. Even neglecting the required time for computing a pair of tangents for regions with a significant number of vertices, the runtime for computing $\mathcal{L}$ remains $\mathcal{O}(n^3)$. This makes this precomputation prohibitively expensive, even before computing the SCP solution. See Section 5 for an experimental evaluation.

▶ **Lemma 3.1.** *Let r_i and r_j be two disjoint compact, convex regions in the plane.*
1. *There are four extremal lines that intersect both r_i and r_j.*
2. *It suffices to consider exterior tangents as potential lines.*

Proof. The first claim is relatively straightforward by considering degrees of freedom and events during continuous modification of a stabbing line.

For the second claim, consider a set of intersected regions $r_1, \ldots, r_s$ (in this order) and an interior tangent between r_{i_1} and r_{i_2} with $i_1 < i_2$; w.l.o.g., let t have positive slope. For any tangential position, we distinguish between regions on the left and right; w.l.o.g., let r_{i_1} be to the left and r_{i_2} to the right of t.

Now rotate t continuously in clockwise direction while maintaining tangential position relative to r_{i_1}. Then all intersections remain intact, until a tangential event with a region $r_{i_3} \neq r_{i_1}$ happens.

We distinguish:

1) If $i_3 > i_1$, then r_{i_3} is to the left, and we have an exterior tangent for r_{i_1} and r_{i_3} that stabs $r_1, \ldots, r_s$.

2) If $i_3 < i_1$, then r_{i_3} is to the right. Then we continue analogously, rotating t counterclockwise around r_{i_3} until we get an event at a region r_{i_4}, with further case distinction.

2.1) If $i_4 > i_3$, then r_{i_3} is to the right, and we have an exterior tangent for r_{i_3} and r_{i_4} that stabs $r_1, \ldots, r_s$.

2.2) If $i_4 < i_3$, then r_{i_4} is to the right. This brings us back to the same situation we had with r_{i_1} and r_{i_2}, but with $i_4 < i_1$, so we can continue in this manner, leading to a sequence $i_1, i_2, i_3, i_4, \ldots$ of event regions. If the current tangential index i_j ever increases, we have identified an exterior tangent; however, the available index set is finite, so a decrease below $i_j = 1$ guarantees an exterior tangent.

◀

Rotating Tangents. A more efficient method for computing a candidate set of tangent lines $\mathcal{L}$ is illustrated in Figure 2 (right). This *Rotating Tangent* (RT) algorithm considers a tangent line that rotates continuously around a compact convex region R_i, and exploits the fact that it intersects another disjoint compact in a contiguous circular arc of directions. This induces a circular arc graph for each of the regions; any maximal clique in this graph corresponds to a maximal subset of intersected regions [20].

Therefore, we can compute all maximal subsets containing a given region in linear time after presorting the events. Computing the tangents and initializing the sweep lines for all polygons takes $\mathcal{O}(n^2)$; presorting and computing all sets during the sweep takes $\mathcal{O}(n^2 \log n)$.

Eliminating Subsets. With either method, the resulting $\mathcal{L}$ may produce a family that contains proper subsets. In principle, these could be eliminated post-construction in worst case $\mathcal{O}(n^5)$; more efficient practical methods (e.g., using k-d trees or other decompositions) could be employed. This turned out to have limited benefit, as some of the methods for the SCP already deal with subsets in a relatively effective manner during their solution process.

4 Solving Set Cover Instances

Now we consider different approaches for finding a subset of $\mathcal{L}$ that covers all regions. Throughout this section, we adopt a standardized notation for the underlying Set Cover problems, where the objective is to cover elements in E by selecting sets from $\mathcal{S}$.

Integer Programming

An Integer Programming formulation is shown in Figure 3; in the worst case, this can result in n^2 sets with 2 elements. Alternatively, we remove all 2-sets from $\mathcal{S}$, i.e., $\mathcal{S}' = \{S_i \mid S_i \in \mathcal{S}, |S_i| > 2\}$ and introduce new binary variables y_j for $j \in E$, see the right side of Figure 3. Constraints are satisfied by choosing a set or its newly introduced variable for covering; the latter option incurs a penalty term of $\frac{1}{2}$ for covering remaining point pairs by lines.

$$\begin{aligned} \min \quad & \sum_{S_i \in \mathcal{S}} x_i & \\ \text{s.t.} \quad & \sum_{\substack{S_i \in \mathcal{S} \\ j \in S_i}} x_i \geq 1 & \forall j \in E \\ & x_i \in \{0,1\} & \forall S_i \in \mathcal{S} \end{aligned}$$

$$\begin{aligned} \min \quad & \sum_{S_i \in \mathcal{S}'} x_i + \frac{1}{2} \sum_{j \in E} y_j & \\ \text{s.t.} \quad & \sum_{\substack{S_i \in \mathcal{S}' \\ j \in S_i}} x_i + y_j \geq 1 & \forall j \in E \\ & x_i \in \{0,1\} & \forall S_i \in \mathcal{S}' \\ & y_j \in \{0,1\} & \forall j \in E \end{aligned}$$

Figure 3 Two possible Integer Programming formulations for PLC and GLC. (Left) Basic set cover IP. (Right) Formulation without sets of size 2.

$$\begin{aligned} \min \quad & \sum_{S_i \in \mathcal{S}} x_i & \\ \text{s.t.} \quad & \bigvee_{\substack{S_i \in \mathcal{S} \\ j \in S_i}} x_i & \forall j \in E \\ & x_i \in \mathbb{B} & \forall S_i \in \mathcal{S} \end{aligned}$$

$$\begin{aligned} \min \quad & 2 \cdot \sum_{S_i \in \mathcal{S}'} x_i + \sum_{j \in E} y_j & \\ \text{s.t.} \quad & \bigvee_{\substack{S_i \in \mathcal{S}' \\ j \in S_i}} x_i \vee y_j & \forall j \in E \\ & x_i \in \mathbb{B} & \forall S_i \in \mathcal{S}' \\ & y_j \in \mathbb{B} & \forall j \in E \end{aligned}$$

Figure 4 Two possible Constraint Programming formulations for PLC and GLC. (Left) Constraint programming formulation. (Right) Formulation without sets of size 2.

Constraint Programming Formulation

The IP can be directly converted into a Constraint Programming formulation, see Figure 4. For a formulation without sets of size 2, we multiply the objective function by a factor of 2 to ensure that the values remain integer.

Large Neighborhood Search

We also tested a Large Neighborhood Search (LNS): Iteratively remove a subset from the current solution until a certain number of elements are uncovered, then solve the restricted set cover problem with an exact solver for the improved IP formulation. In this process, the neighborhood size is adapted to ensure optimal solvability.

5 Experimental Results

Our implementation uses exact number types and predicates and was tested on a workstation with an AMD Ryzen 9 7900 (12×3.7 GHz) CPU and 98GB of RAM[1]. IP solvers are denoted by *IP* (Figure 3 left) and *IP-2SET* (Figure 3 right), CP-SAT solvers as *CP-SAT* (Figure 4 left) and *CP-SAT-2SET* (Figure 4 right). We used *IP-2SET* as the exact solver within the LNS algorithm, denoted by *LNS (IP-2SET)*. To account for the lack of

[1] Source code and data: `https://gitlab.ibr.cs.tu-bs.de/alg/geometric-covering`

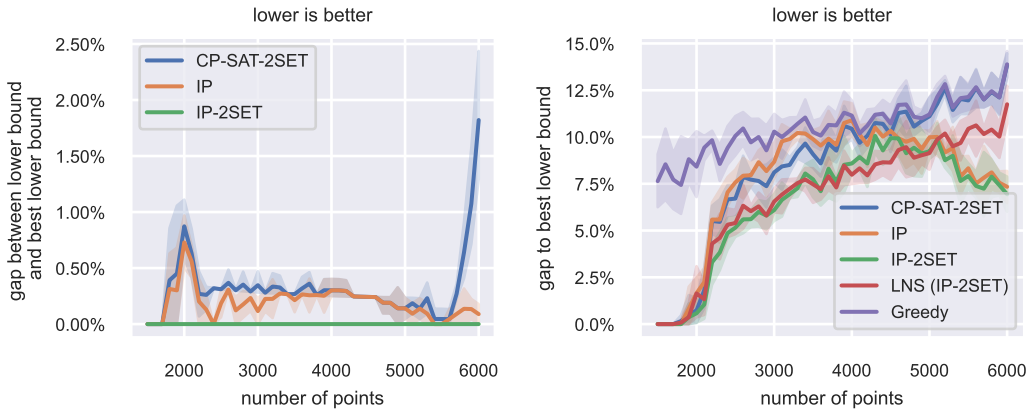

Figure 5 Solvers executed on the *plc_points* instance set with a 600s time limit. (Left) Quality of the lower bounds produced by the different solvers in comparison with the best lower bound. (Right) Upper bound quality (gaps to best lower bound) of all implemented solvers.

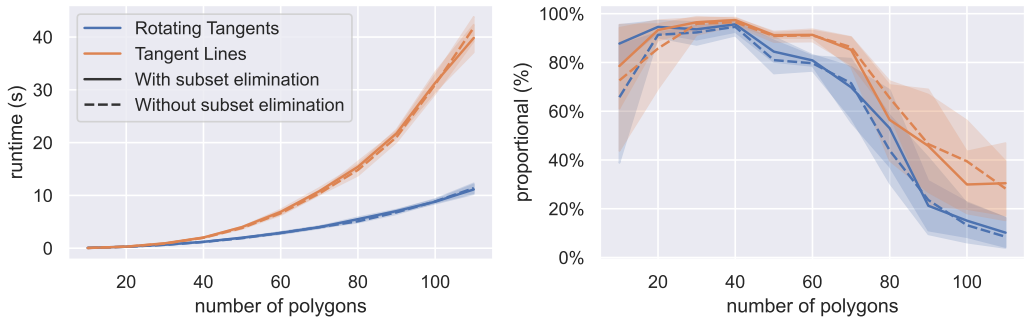

Figure 6 Solvers and line construction algorithms executed on the *glc_polygons_sm* instance set without any time limit (until OPT was found). (Left) Runtime of the RT and TL approach for computing all lines for the set cover problem. (Right) Proportion of time in the full solving process spent during line construction.

publicly available benchmarks, we generated instance sets *plc_points*, *glc_polygons_sm*, *glc_polygons*, *glc_squares* within a fixed-size canvas as follows: For the PLC, we randomly computed lines and chose points on these lines. For the GLC, we placed random point clouds (at locations randomly chosen or according to point locations in TSPLIB [5] instances) and used their convex hull while ensuring no intersections occurred. This yielded several hundred instances; see Figure 1 for examples.

5.1 Point Line Cover

Figure 5 compares the lower bounds from all approaches to the best lower bound found by any method. As the LNS solver can only produce lower bounds for small neighborhoods and *CP-SAT* exceeded the memory limits, they were excluded from this evaluation. Figure 5 shows that the *IP-2SET* can reliably find the best lower bounds of all implemented approaches, even though set cover seems to be suited for SAT-based solvers. The right side of Figure 5 shows that the initial greedy solution already provides reasonably good solutions for all tested instances. *CP-SAT-2SET* performed worse than the IP-based approaches.

Depending on the instance, either *IP-2SET* or the LNS-based approach yielded the best upper bounds, with LNS performing poorly on instances with more than 5000 points.

5.2 General Line Cover

Covering Set Computation

See Figure 6 for a comparison between the two methods for subset computation: RT is considerably faster than the TL approach, despite producing a slightly larger set of candidate lines. Moreover, subset elimination can drastically reduce the number of lines for the SCP solver. However, this has almost no effect on the ensuing SCP computation times.

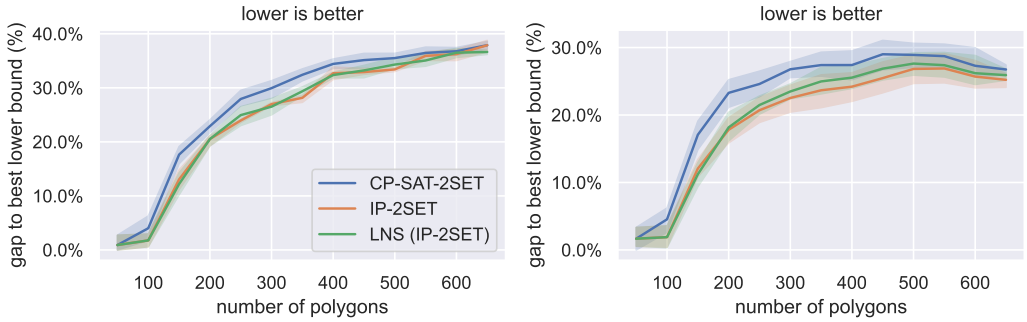

Figure 7 Solvers executed with a 600s time limit. (Left) Upper bound quality (gaps to best lower bound) for convex polygons (*glc_polygons*). (Right) Upper bound quality (gaps to best lower bound) for square instances (*glc_squares*).

Upper bounds

Figure 7 compares the performance of the best solvers from the previous section on two benchmark sets (i) convex polygons of various sizes and (ii) squares, see Figure 1 for examples and solutions. In Figure 7, we can see that *IP-2SET* again beats the other approaches, while *LNS* produces similarly good and sometimes better upper bounds than the integer program. Overall, the gap between the upper and lower bounds is slightly smaller for the unit square instances, and the performance of all approaches is worse than for the PLC.

6 Conclusion

We have shown that geometric covering problems can be practically solved to near optimality for a wide range of instances. A spectrum of further refinements remains to be studied. This includes specialized methods for congruent regions (such as squares or disks, which arise from error bounds for imprecise data), but also higher-dimensional scenarios. As Estivill-Castro et al. [11] showed, there are FPT-type practical approaches for finding PLC solutions with only few lines; it is conceivable that similar ideas can be extended to GLC instances.

References

1 M. Abrahamsen, A. Adamaszek, and T. Miltzow. The Art Gallery Problem is $\exists\mathbb{R}$-complete. *J. ACM*, 69(1):4:1–4:70, 2022.

2 B. Aronov, M. de Berg, J. Cardinal, E. Ezra, J. Iacono, and M. Sharir. Subquadratic algorithms for some 3Sum-hard geometric problems in the algebraic decision-tree model. *Comput. Geom.*, 109:101945, 2023.

3 B. Aronov, E. Ezra, and M. Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM J. Comput.*, 39(7):3248–3282, 2010.

4 I. Baran, E. D. Demaine, and M. Puatracscu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008.

5 B. Bixby and G. Reinelt. TSPLIB, a Traveling Salesman Problem library. *ORSA Journal on Computing*, 3:376–384, 1991.

6 B. Brodén, M. Hammar, and B. J. Nilsson. Guarding lines and 2-link polygons is APX-hard. In *Canadian Conf. Comput. Geometry (CCCG)*, pages 45–48, 2001.

7 V. Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979.

8 M. Dom, M. R. Fellows, and F. A. Rosamond. Parameterized complexity of stabbing rectangles and squares in the plane. In *Intern. Workshop on Algorithms and Computation (WALCOM)*, pages 298–309. 2009.

9 H. Edelsbrunner and M. Sharir. The maximum number of ways to stabbing convex non-intersecting sets in the plane is $2n - 2$. *Discrete & Computational Geometry*, 5(1):35–42, 1990.

10 A. Efrat, S. P. Fekete, J. S. B. Mitchell, V. Polishchuk, and J. Suomela. Improved approximation algorithms for relay placement. *ACM Trans. Algorithms*, 12(2):20:1–20:28, 2016.

11 V. Estivill-Castro, A. Heednacram, and F. Suraweera. Reduction rules deliver efficient FPT-algorithms for covering points with lines. *ACM J. Exp. Algorithmics*, 14, 2009.

12 G. Even, R. Levi, D. Rawitz, B. Schieber, S. M. Shahar, and M. Sviridenko. Algorithms for capacitated rectangle stabbing and lot sizing with joint set-up costs. *ACM Trans. Alg.*, 4(3):34:1–34:17, 2008.

13 S. P. Fekete, K. Huang, J. S. B. Mitchell, O. Parekh, and C. A. Phillips. Geometric hitting set for segments of few orientations. *Theory Comput. Syst.*, 62(2):268–303, 2018.

14 A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.

15 D. R. Gaur and B. Bhattacharya. Covering points by axis parallel lines. In *European Workshop on Computational Geometry (EuroCG)*, pages 42–45, 2007.

16 D. R. Gaur, T. Ibaraki, and R. Krishnamurti. Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem. In *European Symposium on Algorithms (ESA)*, pages 211–219. 2000.

17 P. Giannopoulos, C. Knauer, G. Rote, and D. Werner. Fixed-parameter tractability and lower bounds for stabbing problems. *Comp. Geometry*, 46:839–860, 2013.

18 M. Grantson and C. Levcopoulos. Covering a set of points with a minimum number of lines. In T. Calamoneri, I. Finocchi, and G. F. Italiano, editors, *International Conference on Algorithms and Complexity (CIAC)*, pages 6–17, 2006.

19 R. Hassin and N. Megiddo. Approximation algorithms for hitting objects with straight lines. *Discret. Appl. Math.*, 30(1):29–42, 1991.

20 W.-L. Hsu and K.-H. Tsai. Linear time algorithms on circular-arc graphs. *Information Processing Letters*, 40(3):123–129, 1991.

21 S. Kovaleva and F. C. Spieksma. Approximation algorithms for rectangle stabbing and interval stabbing problems. *SIAM J. Discrete Math.*, 20(3):748–768, 2006.

22 S. Kratsch, G. Philip, and S. Ray. Point line cover: The easy kernel is essentially tight. *ACM Trans. Algorithms*, 12(3):40:1–40:16, 2016.

23 V. A. Kumar, S. Arya, and H. Ramesh. Hardness of set cover with intersection 1. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 624–635, 2000.

24 S. Langerman and P. Morin. Covering things with things. *Discrete & Computational Geometry*, 33(4):717–729, 2005.

25 N. Megiddo and A. Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, 1982.