Covering a Set of Points with a Minimum Number of Lines

Magdalene Grantson Christos Levcopoulos *

Abstract

We consider the minimum line covering problem: given a set S of n points in the plane, we want to find the smallest number l of straight lines needed to cover all n points in S. We show that this problem can be solved in $O(n \log l)$ time if $l \in O(\log^{1-\epsilon} n)$, and that this is optimal in the algebraic computation tree model (we show that the $\Omega(n \log l)$ lower bound holds for all values of l up to $O(\sqrt{n})$). Furthermore, a $O(\log l)$-factor approximation can be found within the same $O(n \log l)$ time bound if $l \in O(\sqrt[4]{n})$. For the case when $l \in \Omega(\log n)$ we suggest how to improve the time complexity of the exact algorithm by a factor exponential in l.

1 Introduction

We consider the minimum line covering problem: given a set S of n points in the plane, we want to find the smallest number l of straight lines needed to cover all n points in S. The corresponding decision problem is: given a set S of n points in the plane and an integer k, we want to know whether it is possible to find k (or fewer) straight lines that cover all n points in S.

Langerman and Morin [7] showed that the decision problem can be solved in $O(nk + k^{2(k+1)})$ time. In this paper we show that the decision problem can be solved in $O(n \log k + (k/2.2)^{2k})$ time.

Kumar *et al.* [6] showed that the minimum line covering problem is APX-hard. That is, unless $P = NP$, there does not exist a $(1 + \epsilon)$-approximation algorithm. In their paper they pointed out that the greedy algorithm proposed by Johnson [5], which approximates the set covering problem within a factor of $O(\log n)$, is the best known approximation for the minimum line covering problem. In this paper we show that a $O(\log l)$-factor approximation for the minimum line covering problem can be obtained in time $O(n \log l + l^4 \log l)$.

We also present an algorithm that solves the line covering problem exactly in $O(n \log l + (l/2.2)^{2l})$ time. This simplifies to $O(n \log l)$ if $l \in O(\log^{1-\epsilon} n)$, and we show that this is optimal in the algebraic computation tree model. That is, we show that the $\Omega(n \log l)$ lower bound holds for all values of l up to $O(\sqrt{n})$. We also suggest more asymptotic improvements for our exact algorithms when $l \in \Omega(\log n)$.

*Dept. of Computer Science, Lund University, Box 118, 221 Lund, Sweden {magdalene,christos}@cs.lth.se

2 Preliminaries

Lemma 1 *Any set S of n points in the plane can be covered with at most $\lceil \frac{n}{2} \rceil$ straight lines.*

Proof. A simple way to show this upper bound is to pick two points at a time, to construct a line through the pair, and then to remove the pair from the set. For the special case when n is odd, we can draw an arbitrary line through the last point. The time complexity of this algorithm is obviously $O(n)$. □

Lemma 2 *If a set S of n points can be covered with k lines (k minimal or not), then: for any subset $R \subseteq S$ of at least $k + 1$ collinear points (i.e., $|R| \geq k + 1$ and $\forall \vec{r}_1, \vec{r}_2, \vec{r}_3 \in R : \vec{r}_1 \neq \vec{r}_2 \Rightarrow \exists \alpha \in \mathbb{R} : \vec{r}_3 = \alpha \cdot (\vec{r}_2 - \vec{r}_1) + \vec{r}_1$), the line through them is in the set of k covering lines.*

Proof. Suppose the line through the points in R was not among the k lines covering S. Then the points in R must be covered with at least $k + 1$ lines, since no two points in R can be covered with the same line. (The only line covering more than one point in R is the one through all of them, which is ruled out.) Hence we need at least $k + 1$ lines to cover the points in R. This contradicts the assumption that S can be covered with k lines. □

Lemma 3 *If a set S of n points can be covered with k lines (k minimal or not), then: any subset of S containing at least $k^2 + 1$ points must contain at least $k + 1$ collinear points.*

Proof. Suppose there is a subset $R \subseteq S$ containing at least k^2+1 points, but not containing $k+1$ collinear points. Then each of the k covering lines must contain at most k points in R. Hence with these at most k covering lines, each containing at most k points, we can cover at most k^2 points. Thus we cannot cover R (nor any superset of R, like S) with the k lines. This contradicts the assumption that S can be covered with k lines. □

Corollary 1 *If in any subset of S containing at least $k^2 + 1$ points we do not find $k + 1$ collinear points, we can conclude that S cannot be covered with k lines.*

Lemma 4 *If a set S of n points can be covered with l lines, but not with $l-1$ lines (i.e., if l is the minimum number of lines needed to cover S) and $k \geq l$, then: if we generate all lines containing more than k points, the total number of uncovered points will be at most $l \cdot k$.*

Proof. Let R be the set of uncovered points in S after all lines containing more than k points have been generated. Since S can be covered with l lines and $R \subseteq S$, R can be covered with l (or fewer) lines. None of the lines covering points in R can cover more than k points in R (as all such lines have already been generated). Hence there can be at most $l \cdot k$ points in R. □

3 General Procedure

Given a set S of n points in the plane, we already know (because of Lemma 1) that the minimum number l of lines needed to cover S is in $\{1, \ldots, \lceil \frac{n}{2} \rceil\}$. In our algorithm, we first check whether $l = 1$, which can obviously be decided in time linear in n. If the check fails (i.e., if the points in S are not all collinear and thus $l \geq 2$), we try to increase the lower bound for l by exploiting Lemmas 2 and 3, which (sometimes) provide us with means of proving that the set S cannot be covered with a certain number k of lines. In the first place, if for a given value of k we find a subset $R \subseteq S$ containing $k^2 + 1$ points, but not containing $k+1$ collinear points, we can conclude that more than k lines are needed to cover S (because of Corollary 1). On the other hand, if we find $k + 1$ collinear points (details of how this is done are given below), we record the line through them (as it must be among the covering lines due to Lemma 2) and remove from S all points covered by this line. This leads to a second possible argument: If by repeatedly identifying lines in this way (always choosing the next subset R from the remaining points), we record k lines while there are points left in S, we can also conclude that more than k lines are needed to cover S.

We check different values of k in increasing order (the exact scheme is discussed below), until we reach a value k_1, for which we fail to prove (with the means mentioned above) that S cannot be covered with k_1 lines. On the other hand, we can demonstrate (in one of the two ways outlined above) that S cannot be covered with k_0 lines, where k_0 is the largest value smaller than k_1 that was tested in the procedure. At this point we know that $l > k_0$.

Suppose that when processing S with $k = k_1$, we identified m_1 lines, $m_1 \leq k_1$. We use a simple greedy algorithm to find m_2 lines covering the remaining points. (Note that m_2 may or may not be the minimum number of lines needed to cover the remaining points. Note also that $m_2 = 0$ if there are no points left to be covered.) As a consequence we know that S can be covered with $m_1 + m_2$ lines (since we have found such lines) and thus that $k_0 < l \leq m_1 + m_2$. We show in [3] that $m_1 + m_2 \in O(l \log l)$ and thus that with the $m_1 + m_2$ lines we selected we obtained an $O(\log l)$ approximation of the optimum.

In a second step we may then go on and determine the exact value of l by drawing on the found approximation (See the full version of paper in [3] for details).

Our proposed approximate and exact algorithms to solve the minimum line covering problem use the following two already known algorithms as subroutines:

1. An algorithm proposed by Guibas *et al.* [4], which finds all lines containing at least $k + 1$ points in a set S of n points in time $O\left(\frac{n^2}{k+1} \log \frac{n}{k+1}\right)$.

2. An algorithm proposed by Langerman and Morin [7], which takes as input a set S of n points and an integer k, and outputs whether S can be covered with k lines in $O(nk + k^{2(k+1)})$ time.

3.1 Approximation for Minimum Line Covering

Lemma 5 *We can approximate the minimum line covering problem within a factor of $O(\log l)$ in $O(n \log l)$ time if $l \leq \sqrt[4]{n}$.*

Theorem 6 *We can approximate the minimum line covering problem within a factor of $O(\log l)$ in time $O(n \log l + l^4 \log l)$.*

Corollary 2 *We can approximate the minimum line covering problem within a factor of $O(\log l)$ in $O(n \log l)$ time if $l \in O(\sqrt[4]{n})$.*

See the full paper [3] for the proofs of the above results.

3.2 Exact Minimum Line Covering

Theorem 7 *The minimum line covering problem can be solved exactly in $O(n \log l + l^{2l+2})$ time. In particular, if $l \in O(\log^{1-\epsilon} n)$, the minimum line covering problem can be solved in $O(n \log l)$ time.*

Theorem 8 *Given a set S of n points in the plane and an integer k, we can answer whether it is possible to find k lines that cover all the points in the set in $O(n \log k + k^{2k+2})$ time.*

See the full paper [3] for the proofs for the above Theorems.

3.3 Producing the optimal set of lines

We also remark that after computing the optimal number of lines l, we can also produce the actual lines covering the input point set within the same time bounds. One way is to first use the algorithm proposed by Guibas *et al.* [4] to produce lines covering at least $l+1$ points. Let l' denote the number of lines covering at least $l+1$ points and n' the number of points left to be covered. We observe that at least one of the remaining $l-l'$ lines cover at least $\frac{n'}{l-l'}$ points. Let a line be called a *candidate* line if it covers at least $\frac{n'}{l-l'}$ points. Let us now compute the first candidate line: If $n' \leq 2(l-l')$ then any line covering at least two points can be included in the optimal solution. Otherwise, we tentatively (temporarily) remove the points covered by it and call Langerman and Morin's algorithm [7] to see whether the remaining points can be covered by $l-l'-1$ lines. Clearly, the candidate can be included in the optimal solution if and only if the answer is yes. Let n_r denote the remaning points to be covered and l_n denote the number of lines needed to cover n_r. We repeat the above algorithm and update n_r and l_n accordingly after each construction of a candidate line.

To calculate the time bound we show that there are at most $\frac{3}{2} \cdot (l-l')^2$ candidate lines. Any point can be covered by no more than $\frac{3}{2} \cdot (l-l')$ candidate lines. (The factor $\frac{3}{2}$ comes from the extreme case when $l-l' = \frac{n'}{3}$, so that each candidate line covers only three points and the point is covered by $\frac{n'-1}{2}$ candidates.) Hence, if we sum for each point the number of candidates it is covered by, we thus get an upper bound of $n' \cdot \frac{3}{2}(l-l')$. But we observe that this sum equals the sum we obtain by adding for each candidate line the number of points it covers. Since each candidate line covers at least $\frac{n'}{l-l'}$ points, the number of candidate lines cannot be larger than $(n' \cdot \frac{3}{2}(l-l'))/\frac{n'}{l-l'}$ and hence not larger than $\frac{3}{2}(l-l')^2$. Therefore we call Langerman and Morin's algorithm [7] at most $\frac{3}{2}(l-l')^2$ times before we produce one more optimal line. In subsequent calls to their algorithm, the number of optimal lines to be produced gets smaller and hence the time complexity gets smaller each time by at least a constant factor, since it is exponential in the number of optimal lines. This results in a geometric progression of the time complexity. Therefore the worst-case bound for the first call asymptotically dominates all subsequent calls.

3.4 Improving the time bound when $l \in \Omega(\log n)$

Theorem 9 *For any input set of n points, it can be decided whether there is a set of lines of cardinality at most k covering the n points in time $O(n \log k + (\frac{k}{2.2194...})^{2k})$. Moreover, an optimal set of covering lines with minimum cardinality l can be produced in time $O(n \log l + (\frac{l}{2.2194...})^{2l})$*

Proof. See the full paper [3] of the proof. □

4 Lower Bound

In this section we give a lower bound on the time complexity for solving the minimum line cover problem. We make the assumption that the minimum number of lines l needed to cover a set S of n points is at most $O(\sqrt{n})$. (For larger values of l our lower bound may not be very interesting, since the best known upper bounds on the time complexity of the minimum line cover problem are exponential anyway.)

The main result we prove in this section is as follows:

Theorem 10 *The time complexity of the minimum line cover problem is $\Omega(n \log l)$ in the algebraic decision tree model of computation.*

We prove Theorem 10 with a reduction from a special variant of the general set inclusion problem [1], which we call the k-Array Inclusion problem. Set inclusion is the problem of checking whether a set of m items is a subset of the second set of n items with $n \geq m$. Ben-Or [1] showed a lower bound of $\Omega(n \log n)$ for this problem using the following important statement.

Statement 1 *If YES instances of some problem Π have N distinct connected components in $\Re^n$, then the depth of the real computation tree for this problem is $\Omega(\log N - n)$.*

Applying Statement 1 to the complement of Π, we get the same statement for NO instances. We define the k-array inclusion problem as follows:

Definition 1 **k-Array Inclusion Problem:** *Given two arrays $A[1 \ldots k]$ of distinct real numbers and $B[1 \ldots m]$ of (not necessarily distinct) real numbers, $k \leq m$, $m+k=n$, determine whether or not each element in $B[1 \ldots m]$ belongs to $A[1 \ldots k]$.*

Corollary 3 *Any algebraic computation tree solving k-array inclusion problem must have a depth of $\Omega(n \log k)$.*

Proof. This lower bound can be shown in a corresponding way as the lower bound for the set inclusion problem [1]. As already pointed out by Ben-Or, any computational tree will correctly decide the case when $A[1 \ldots k] = (1 \ldots k)$. The number of disjoint connected components, N, for YES instances of the k-array inclusion problem is k^m. This is because, in order to create a YES instance, for each element m_i in $B[1 \ldots m]$, there are k choices concerning which of

the k fixed elements in $A[1 \dots k]$, m_i could be equal to. Since these choices are independent for each m_i, the total number of YES-instances becomes k^m. Applying Statement 1, we get a lower bound of $\Omega(m \log k)$, which is also $\Omega(n \log k)$, since $m > \frac{n}{2}$. □

To establish the lower bound in Theorem 10 for the minimum line cover problem, we convert (in linear time) the input of the k-array inclusion problem into a suitable input to the minimum line cover problem as follows: Each real number a_i, $1 \leq i \leq k$, in the array $A[1 \dots k]$ becomes k points with coordinates (a_i, j) (in total we obtain k^2 points), $1 \leq j \leq k$ and each real number b_j in the array $B[1 \dots m]$, $1 \leq j \leq m$, becomes a point with coordinates $(b_j, -j)$, all points are in two dimensional space. None of the constructed sets of $n = k + m$ points coincide. If we use any algorithm for the minimum line cover problem to solve the constructed instance, the output will be a set of lines covering these points. To obtain an answer to the k-array inclusion problem, we check whether the total number of lines, denoted by l, obtained for the minimum line cover problem is greater than k. If $l = k$ then each element in $B[1 \dots m]$ belongs to $A[1 \dots k]$, otherwise at least one element in $B[1 \dots m]$ does not belong to $A[1 \dots k]$. Since the k-array inclusion problem requires $\Omega(n \log k)$ time, it follows that the minimum line cover problem requires $\Omega(n \log k)$ time as well.

According to this construction, if it would be possible to compute the number l in $o(n \log l)$ time, for some $l = O(\sqrt{n})$, then it would also be possible to solve the k-array inclusion problem in time $o(n \log k)$ for the case when $k = l$, which would contradict our lower bound for the k-array inclusion problem.

Acknowledgment: The authors wish to thank Dr. Christian Borgelt for his detailed reading and comments on the paper.

References

[1] M. Ben-Or. Lower Bounds for Algebraic Computation Trees. *Proc. 15th Annual ACM Symp. on Theory of Computing*, 80–86. ACM Press, New York, NY, USA 1983

[2] H. Edelsbrunner, L. Guibas and J.Stolfi. Optimal Point Location in a Monotone Subdivision. *SIAM J. Comput.* 15:317–340. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA 1986

[3] M. Grantson and C. Levcopoulos. Covering a Set of Points with a Minimum Number of Lines. `http://www.cs.lth.se/~magdalene/lines.pdf`

[4] L. Guibas, M. Overmars, J. Robert. The Exact Fitting Problem in Higher Dimensions. *Computational Geometry: Theory and Applications*, 6:215–230. 1996

[5] D. Johnson. Approximation Algorithms for Combinatorial Problems. *J. of Comp. Syst. Sci.* 9:256-278. 1974

[6] V. Kumar, S. Arya, and H. Ramesh. Hardness of Set Cover With Intersection 1. *Proc. 27th Int. Coll. Automata, Languages and Programming,* LNCS 1853:624–635. Springer-Verlag, Heidelberg, Germany 2000

[7] S. Langerman and P. Morin. Covering Things with Things. *Proc. 10th Annual Europ. Symp. on Algorithms (Rome, Italy)*, LNCS 2461:662–673. Springer-Verlag, Heidelberg, Germany, 2002

[8] N. Megiddo and A. Tamir. On the Complexity of Locating Linear Facilities in the Plane. *Operation Research Letters* 1:194–197. 1982

[9] N. Sarnak, and R.E. Tarjan. Planar Point Location Using Persistent Search Tree. *Comm. ACM* 29:669-679. ACM Press, New York, NY, USA 1986