An Optimal Algorithm for Approximating a Set of Rectangles by Two Minimum Area Rectangles*

Bruno Becker† Paolo Giulio Franciosa‡ Stephan Gschwind†
Thomas Ohler† Gerald Thiemt† Peter Widmayer†

Abstract

In this paper we face the problem of computing a conservative approximation of a set of isothetic rectangles in the plane by means of a pair of enclosing isothetic rectangles. We propose an $O(n \log n)$ time algorithm for finding, given a set M of n isothetic rectangles, a pair of isothetic rectangles (s, t) such that s and t enclose all rectangles of M and $area(s) + area(t)$ is minimal. Moreover we prove an $O(n \log n)$ lower bound for the one-dimensional version of the problem.

1 Introduction

Computing approximated, concise representations of complex shapes is a standard problem in computer graphics, pattern recognition and robotics. [1] shows how a convex polygon can be approximated by means of circles, rectangles or k-gons. An algorithm for computing a pair of parallel circumscribed and inscribed rectangles for a convex polygon with n vertices is described in [6]; its complexity is $O(\log^3 n)$ if vertices are given in a sorted array. In [3] a convex polygon with n vertices is approximated by means of a minimum area enclosing triangle in time $O(n)$, starting from the sorted list of the vertices; a very simple $O(n \log n)$ time algorithm for computing the minimum area enclosing circle is given in [7].

In this paper we face the particular approximation problem in which the object to be approximated is a set of isothetic rectangles (which could be the bounding boxes of parts that form an object), and we want to approximate the object by means of a pair of isothetic rectangles enclosing it. Our goal is to find, among all the pairs (s, t) of enclosing rectangles, one for which the sum of the areas of s and t is minimal.

We use the following notations in this paper: A *closed one-dimensional interval* $i = [i.l, i.r] \subseteq \mathcal{R}$, $i.l \in \mathcal{R}$, $i.r \in \mathcal{R}$, is given by its left and right boundaries. In the rest of this paper, *interval* always

*Work partially supported by grants no. Wi810/2–5 from the Deutsche Forschungsgemeinschaft, by the ESPRIT II Basic Research Actions Program of the European Community, Working Group "Basic GOODS", and by the Italian PFI National Project "Obiettivo Multidata".
†Institut für Informatik, Universität Freiburg, Rheinstraße 10–12, D–7800 Freiburg, Germany.
‡Istituto di Analisi dei Sistemi ed Informatica, viale Manzoni 30, I–00185 Roma, Italy.

denotes a *closed interval*. The *length* of a one-dimensional interval i is defined as $length(i) = i.r - i.l$. An *isothetic rectangle* in the plane, i.e. a two-dimensional interval, is defined as the cartesian product of two one-dimensional intervals. The *area* of an isothetic rectangle is given by the product of the lengths of its one-dimensional intervals. In the following, we use the term *rectangle* instead of *isothetic rectangle*. We define the union (intersection) of two intervals as the closure of the union (intersection) of their interiors; these are the regularized operators of [8].

Now we define our approximation problem more formally as follows: Given a set M of rectangles in $\mathcal{R}^2$, find a pair of rectangles s and t such that

$$s \cup t \supseteq \bigcup_{r \in M} r,$$

minimizing $area(s) + area(t)$.

We provide an $O(n \log n)$ worst case time and $O(n \log \log n)$ space algorithm for solving this problem, and we prove an $O(n \log n)$ lower time bound by reduction to the *maximum gap* problem [5].

The algorithm distinguishes three types of enclosing rectangle pairs, on the basis of the disjointness of s and t and of their relative position. The optimal solution is computed as the best one among the minimum area solutions for each type. The best solutions for two of the types directly can be found in $O(n \log n)$ time and $O(n)$ space, while the best solution for the third type is found by applying the compact interval tree datastructure described in [2], resulting in an $O(n \log n)$ worst case time and $O(n \log \log n)$ space performance.

2 A Lower Time Bound

We show that in the algebraic decision tree model [5] $\Omega(n \log n)$ worst case time is needed to find the minimum area solution for our problem. We prove this lower bound in the one-dimensional case; it trivially extends to the two-dimensional problem.

Lemma 2.1 *Given a set M of n one-dimensional intervals $i_1, i_2, \ldots, i_n$, $\Omega(n \log n)$ worst case time is needed for finding two intervals s and t such that*

$$s \cup t \supseteq \bigcup_{j=1}^{n} i_j,$$

minimizing $length(s) + length(t)$.

Proof. We exploit an $\Omega(n \log n)$ lower bound for the *maximum gap* problem [5], by showing that the maximum gap problem can be reduced to the one-dimensional interval approximation problem of Lemma 2.1.

The maximum gap problem is defined in [5] as follows:
Given a set $A=\{a_1, a_2, \ldots, a_n\}$ of n reals, find the maximum γ such that there exist i and j for

Figure 1: Reduction from *maximum gap* to interval approximation.

which $a_j - a_i = \gamma$ and $\forall k \;\; a_k \leq a_i$ or $a_j \leq a_k$.
Let $max = \max_{i=1}^{n}(a_i)$ and $min = \min_{i=1}^{n}(a_i)$.
A simple argument shows that for $\epsilon = \frac{max - min}{n}$, we certainly have $\gamma > \epsilon$. Let us now define $M = \{[a_i, a_i + \epsilon] \mid i = 1, \ldots, n\}$. Note that there is a gap of size at least $\delta = max - min - x + 2\epsilon$ in A, $\delta > \epsilon$, if and only if the pair (s, t) is a feasible solution for M and $length(s) + length(t) = x$ (see Figure 1). Hence maximizing δ corresponds to finding the minimum of $length(s) + length(t)$.
$\square$

3 An Optimal Algorithm

We separately look for solutions of three different types, finding the minimum area solution for each type, and then we select the best one.

For the given set M of rectangles let $m = box(M)$ denote the bounding box of M, i.e., the minimum rectangle containing M. Since $s \cup t$ must enclose all rectangles in M and $area(s)+area(t)$ has to be minimal, for each horizontal (vertical) side of m there must exist a horizontal (vertical) side of s or t having the same coordinate value.

We disregard reflections, rotations and the exchange of s with t. W.l.o.g. let s have at least as many sides in common with m as t; of course, s must have at least two such sides.

We can therefore distinguish different types of solutions, depending on the sides of s that touch sides of m.

We partition the solutions into the following three types (see Figure 2 for an illustration):

1. s has at least two adjacent sides on the boundary of m, and s and t have no common interior points;

Figure 2: Different types of solutions.

2. s has at least two adjacent sides on the boundary of m, and s and t have some common interior point;
3. s has no two adjacent sides on the boundary of m.

We describe in the following how the best solution for each type is computed.

3.1 Type 1 Solutions

For Type 1, w.l.o.g. s and t can be separated by a vertical line, and s lies to the left of t. We show how the minimum area solution (s,t) can be computed for this type of solution.

Given a fixed vertical line $x=v$, let $l(M,v)$ denote the set of rectangles that intersect the left halfplane $H_l(v)=\{(x,y)|x\leq v\}$, and let $r(M,v)$ denote the set of rectangles that intersect the right halfplane $H_r(v)=\{(x,y)|x\geq v\}$.

Each value v defines a feasible solution (s,t), with $s=box(l(M,v))\cap H_l(v)$ and $t=box(r(M,v))\cap H_r(v)$.

Lemma 3.1 *If (s,t) is a feasible solution defined by a value v that is not the abscissa of the side of a rectangle, then there exists an abscissa v' of a rectangle side defining a solution (s',t') such that*

$$area(s')+area(t')\leq area(s)+area(t)\ .$$

Figure 3: Proof of Lemma 3.1.

Proof. Let a be the largest abscissa of a rectangle side to the left of v and b the smallest abscissa of a rectangle side to the right of v (see Figure 3).

We will distinguish two cases:

- no rectangle intersects the line $x = v$: in this case the abscissa a (or b) defines the same solution as v;
- the line $x = v$ intersects some rectangles: let h_s and h_t be the height of respectively s and t. If $h_s \geq h_t$, abscissa a defines a solution whose area is

  $$area(s) + area(t) - (h_s - h_t) \cdot (v - a) \leq area(s) + area(t).$$

  Similarly, if $h_s < h_t$, then abscissa b defines a solution with no larger area.

□

Lemma 3.1 allows us to only evaluate solutions defined by vertical lines with the same abscissa as the right side or the left side of some rectangle in M, since other solutions cannot have a smaller area.

The *sweep line* algorithm we propose for this task starts at the left end of M. s (resp. t) is the bounding box of all parts of rectangles to the left (resp. right) of the sweep line. s is initialized to the empty rectangle, while $t = m = box(M)$. The line sweep scans each left and right side of all rectangles in ascending order of abscissa values. The left side of s is fixed at the left side of m and the sweep line defines the right side of s. The coordinate of the upper side of s is the highest coordinate of all upper sides of the set of rectangles whose left sides are to the left of the sweep line. Because rectangles are only added to this set while moving the sweep line from left to right, the upper side of s moves monotonously upwards. A similar argument holds for the lower side of s, which moves monotonously downwards.

If the sweep line intersects some rectangles, the left side of t is defined by the sweep line. Otherwise the left side of t is defined by the lowest coordinate of all left sides of the set of rectangles, whose left sides are not left of the sweep line. The right side of t is fixed at the right side of m, the upper side moves downwards and the lower side upwards, and the left side moves from left to right as the sweep line proceeds.

Because all sides of s and t are either fixed or move monotonously in one direction (of course not all in the same direction), we maintain rectangles and coordinates in the following doubly linked lists:

- One (unsorted) list of all rectangles
- Three sorted lists of left, top and bottom coordinates of sides of the rectangles, the *t-lists*
- One sorted list of left and right rectangle coordinates, the sweep line halting points.

Each element of a sorted list is linked with its rectangle element of the rectangle list by pointers in both directions.

The worst case time needed for building all lists and linking them equals the cost of sorting, i.e. $O(n \log n)$. One step in the line sweep can be reflected in the lists by a change of constant cost. The left, bottom and top coordinates of t can always be found in the head elements of *t-lists*. Therefore the total time needed after preprocessing is $O(n)$. Hence the following theorem holds:

Theorem 3.2 *The overall worst case time complexity to compute an optimal solution of Type 1 is $O(n \log n)$.*

3.2 Type 2 Solutions

A polygon is *orthoconvex* if its intersection with any horizontal or vertical line is either a line segment or is empty [4]. The *orthoconvex hull* of a set of rectangles is the smallest orthoconvex polygon which covers all the rectangles. The algorithms for computing solutions of Types 2 and 3 basically use the boundary of the orthoconvex hull of a given set M of rectangles, *boch(M)* for short. Let N, S, E, W be the four compass directions. We call the northernmost edge of *boch(M) N-edge*. E-edge, S-edge and W-edge are defined accordingly. If there exist edges between the N-edge and the W-edge, we call this part of *boch(M)* *NW-staircase*(M) or *NW-stairs*(M) for short. NE-, SE- and SW-staircase(M) are defined correspondingly (see Figure 4).

The NE(SE/SW/NW)-staircase of a set of rectangles can be determined in $O(n \log n)$ worst case time, as described in [5].

For Type 2 w.l.o.g. we will only consider solutions in which the lower-left vertex of s coincides with the lower-left vertex of $box(M)$ and the upper-right vertex of t coincides with the upper-right vertex of $box(M)$.

Figure 4: NE(NW/SW/SE)-staircase of a set of rectangles.

We first prove that the minimum area solutions for Type 2 can be found by exploring two opposite staircases of M.

Lemma 3.3 *If (s,t) is a minimum area solution of Type 2 for M, then the boundaries of s and t intersect NW-stairs(M) in the same concave vertex; the same is true for SE-stairs(M).*

Proof. If (s,t) is a solution of Type 2, then the upper side of s and the left side of t both intersect *NW-stairs*(M).

Suppose now that the upper side of s and the left side of t do not intersect in a concave vertex in *NW-stairs*(M): in this case the upper side of s can be shifted downwards and/or the left side of t can be moved rightwards, obtaining a better solution; hence (s,t) is not a minimum area solution.

The same argument applies to the right side of s and the lower side of t, with respect to *SE-stairs*(M) (see Figure 5).

It is simple to see that the only case in which the left side of t and the right side of s cannot be shifted to a concave vertex occurs when the solution becomes of Type 1. The same argument holds for the top side of s and the bottom side of t. $\square$

Each solution of interest is therefore identified by a pair of concave vertices (p,q), with $p \in$ *NW-stairs*(M) and $q \in$ *SE-stairs*(M).

Before stating the algorithm for finding the minimum area solution for Type 2 we must further for each concave vertex $p \in$ *NW-stairs*(M) reduce the set of vertices explored in *SE-stairs*(M). Given $p = (p_x, p_y) \in$ *NW-stairs*(M), p defines a set of feasible vertices in *SE-stairs*(M), that will be named *active*(M,p). This is the set of concave vertices $q = (q_x, q_y)$ with $q_x > p_x$ and $q_y < p_y$ (see Figure 6).

The *convex hull* of a set of points P is the smallest convex polygon covering all the points. The *upper boundary* is the upper part of the boundary of the convex hull between an extreme left

Figure 5: Proof of Lemma 3.3.

Figure 6: An active interval for p in *SE-stairs*.

Figure 7: The point with minimum area is on the convex hull.

and an extreme right point on the boundary. By $ubvert(P)$ we denote the vertices of the upper boundary of P.

We will show that only vertices of $ubvert(active(M, p))$ must be taken into account for the computation of an optimal solution of Type 2 (see Figure 7).

Lemma 3.4 *If (s, t) is a minimum area solution of Type 2 identified by vertices (p, q), with $p \in NW\text{-}stairs(M)$, then $q \in ubvert(active(M, p))$.*

Proof. Let w and h be respectively the width and the height of $box(M)$, and let $p = (p_x, p_y)$ and a vertex $q' \in active(M, p)$, $q' = (q'_x, q'_y)$; (p, q') identifies a solution with area (see Figure 7)

$$A_{p,q'} = q'_x \cdot p_y + (w - p_x) \cdot (h - q'_y) \ .$$

The set of points $v = (v_x, v_y)$ such that $A_{p,v} = A_{p,q'}$ is a straight line l whose equation is

$$v_y - q'_y = \frac{p_y}{(w - p_x)} \cdot (v_x - q'_x) \ .$$

The minimum area solution (with fixed p) is given by a vertex q in $active(M, p)$ through which a line parallel to l is closest to p.

The claim is now proved, since this can be seen as a two-dimensional linear programming problem whose admissible region is the convex hull of $active(M, p)$; and it is well known that the optimal solution can always be chosen on a vertex of the admissible region. □

A set of Type 2 solutions which contains the optimal one can be found by maintaining $ubvert(active(M, p))$ under insertions and deletions of points which enter and leave the set of active vertices while p moves along concave vertices in $NW\text{-}stairs(M)$ and searching an optimal vertex $q \in ubvert(active(M, p))$ for each fixed p.

We just sketch an algorithm for finding this set of solutions:

- set p to the leftmost concave vertex in *NW-stairs*(M);
- while p is in *NW-stairs*(M) do
  - set $u = ubvert(active(M, p))$;
  - find the optimal vertex q on u;
  - evaluate the solution given by (p, q);
  - set p to the next concave vertex in *NW-stairs*(M).

While p moves rightwards on *NW-stairs*(M), both extremes of $active(M, p)$ can only move rightwards. Hence the total number of deletions and insertions while p scans the whole *NW-stairs*(M) is $O(n)$.

So all we have to do is to maintain the *ubvert* under the following operations:

- insertion of vertices to the right of the current set;
- deletion of the leftmost vertex in the set.

Because *SW-stairs* is a simple path, it is possible to apply the concept of *Compact Interval Trees*, proposed in [2]. With this data structure, which needs $O(n \log \log n)$ space, it is possible to rebuild the convex hull of the active vertices after each insertion or deletion in time $O(\log n)$. A compact interval tree allows binary search in such a way, that an optimal vertex q for a given vertex p can be found in time $O(\log n)$ after the convex hull is computed.

Theorem 3.5 *The overall worst case time complexity to compute an optimal solution of Type 2 is $O(n \log n)$.*

3.3 Type 3 Solutions

We first prove that, as at Type 2 solutions, the minimum area solutions of this type can be found by exploring two opposite staircases of M.

Lemma 3.6 *If (s, t) is a minimum area solution of Type 3 for M, with s having the width of $box(M)$ and t having the height of $box(M)$, then*

- *the top side of s and the left side of t intersect in a point of NW-$stairs(M)$, and the bottom side of s and the right side of t intersect in a point of SE-$stairs(M)$; or*
- *the top side of s and the right side of t intersect in a point of NE-$stairs(M)$, and the bottom side of s and the left side of t intersect in a point of SW-$stairs(M)$.*

Figure 8: Proof of Lemma 3.6.

Proof. It is immediate to see that if (s,t) is a solution of Type 3 then the two vertical (horizontal) sides of s (t) intersect the four staircases of M (see Figure 8).

We can now prove the claim by reduction to the absurd. We choose a side of one of the two solution rectangles, w.l.o.g. the right side of t, and we show that at least one of the intersections of this side with the top and the bottom side of s must lie on a staircase of M. Otherwise, (s,t) could not be a minimum area solution, since the right side of t could be shifted to the left, obtaining a better solution. The claim directly follows. $\square$

Moreover, it is simple to see that points identifying minimum area solutions must be points of the grid defined by the extremal points of rectangles in M, and there are at most $O(n)$ of such points on each of the staircases.

So the minimum area solution is identified by two *grid* points on opposite staircases. Our algorithm will, w.l.o.g., sequentially scan these points in NW-$stairs(M)$, and for each of them will directly find the point in SE-$stairs(M)$ giving the best possible solution.

For each grid point p in NW-$stairs(M)$ all the feasible solutions are identified by one of the *active* grid points q in SE-$stairs(M)$, i.e. the points contained in the interval defined by "reflecting" on NE-$stairs(M)$ and on SW-$stairs(M)$ the horizontal and vertical rays from p (see Figure 9).

The algorithm for finding a minimum area solution of Type 3 is sketched below.

- set p to the leftmost point in NW-$stairs(M)$;
- build the set of points in the active interval corresponding to p;
- while p is in NW-$stairs(M)$ do:
  - find point q defining the minimum area solution for p;

Figure 9: Area of solutions of Type 3.

- evaluate the solution defined by (p, q);
- move p to the next point on the grid in NW-$stairs(M)$ and update the active interval.

Let us now evaluate the area $A(p, q)$ of the solution identified by a given pair of points (p, q), $p = (p_x, p_y)$, $q = (q_x, q_y)$, respectively on the NW-$stairs(M)$ and SE-$stairs(M)$. Let w and h be respectively the width and the height of $box(M)$.

The following equality holds:

$$\begin{aligned} A(p, q) &= (p_y - q_y)w + (q_x - p_x)h \\ &= p_y w - p_x h + q_x h - q_y w \end{aligned}$$

Hence minimizing $A(p, q)$ for a fixed p corresponds to minimizing $q_x h - q_y w$; this value does not depend upon p.

It is important for the efficiency of our algorithm that while point p in NW-$stairs(M)$ moves to the right, the corresponding interval of active points on SE-$stairs(M)$ only can move to the left. Hence each point in SE-$stairs(M)$ can be inserted into and deleted from the set of active points just once. Also, the reflection points for p_x on SW-$stairs(M)$ and for p_y on NE-$stairs(M)$ move along in one direction respectively, namely from top to bottom and from right to left.

The set of active points can be maintained in a partially ordered structure, say a heap, according to the values $q_x h - q_y w$. This allows us to perform detection of minimum in constant time, and insertion and deletion of points in $O(\log n)$ worst case time (deletion will be performed by using a pointer to the node representing a point in the heap).

Hence we can state the following:

Theorem 3.7 *The overall worst case time complexity for computing an optimal solution of Type 3 is $O(n \log n)$.*

4 Conclusion and Further Work

We presented an algorithm for finding, given a set of n isothetic rectangles in the plane, a pair of isothetic rectangles whose union encloses the given rectangles, such that the sum of the areas of the two rectangles is as small as possible.

The pair of minimum area enclosing rectangles can be seen as a conservative approximation for the given set of rectangles, which, depending on the shape of the set of rectangles, can be much closer to these than the simple bounding box, but still requires only constant size storage. Hence this approach can be useful in cases where we are looking for a concise representation of complex objects, as in the design of spatial data structures.

The solutions are partitioned into three types, and the algorithm works by finding the minimum area solution for each type. Its worst case running time is $O(n \log n)$, and it needs $O(n \log \log n)$ space.

Currently we are investigating the possibility of extending the same approach to the approximation of objects other than rectangles, the extension to objects in higher dimension, and the approximation by a higher, but fixed, number of objects.

References

[1] H. Alt, J. Blomer, M. Godau, and H. Wagener. Approximation of convex polygons. In *Automata, Languages and Programming (Proc. of the 17th ICALP, Univ. of Warwick, England, July 1990), Lecture Notes in Computer Science 443*, pages 703–716, 1990.

[2] L. Guibas, J. Hershberger, and J. Snoeyink. Compact interval trees: A data structure for convex hulls. *International Journal of Computational Geometry & Applications*, 1:1–22, 1991.

[3] J. O'Rourke, A. Aggarwal, S. Maddila, and M. Baldwin. An optimal algorithm for finding minimal enclosing triangles. *Journal of the Algorithms*, 7:258–269, 1986.

[4] Th. Ottmann, E. Soisalon-Soininen, and D. Wood. On the definition and computation of rectilinear convex hulls. *Information Sciences*, 33:157–171, 1984.

[5] F. P. Preparata and M. I. Shamos. *Computational Geometry, an Introduction.* Springer-Verlag, New York, 1985.

[6] O. Schwarzkopf, U. Fuchs, G. Rote, and E. Welzl. Approximation of convex figures by pairs of rectangles. Report B 89-15, Freie Universitaet Berlin, Department of Mathematics, 1989.

[7] S. Skyum. A simple algorithm for computing the smallest enclosing circle. Report DAIMI 314, Aarhus University, Computer Science Department, 1990.

[8] R. B. Tilove. Set membership classification: A unified approach to geometric intersection problems. *IEEE Trans. on Computers*, pages 874–883, 1980.