Optimal Polygon Placement by Translation[1]

Sudebkumar Prasant Pal
Department of Computer Science and Engineering
Indian Institute of Technology
Kharagpur 721302, INDIA

Bhaskar Dasgupta
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455-0159, USA

C. E. Veni Madhavan
Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560012, INDIA

Abstract

Let M be an m-sided simple polygon and N be an n-sided polygon with holes. In this paper we consider the problem of computing the *feasible* region i.e., the set of all placements by translation of M so that M lies inside N without intersecting any hole. First we propose an $O(mn^2)$ time algorithm for computing the feasible region for the case when M is a monotone polygon. Then we consider the general case when M is a simple polygon and propose an $O(m^2n^2)$ time algorithm for computing the feasible region. Both algorithms are optimal upto a constant factor.

Keywords

Polygon placement, feasible region, polygon containment, optimal algorithms.

CR Categories : F2.2 I3.5

1 Introduction

Let M be a simple polygon and N be a polygon with holes. In this paper we consider the problem of computing the *feasible region* i.e., the set of all placements by translation of M so that M lies within N without intersecting any hole. Computing the feasible region has applications in planning translational motion of M inside N [8]. By checking whether the feasible region is empty, it is possible to decide whether M can be translated to fit within N (i.e. M can be *contained* in N). This has applications in stockcutting and inspection problems [2,5], and in VLSI.

Computing the feasible region of an m-sided polygon M inside an n-sided polygon N with holes, is a well studied problem. Baker et al. [2], proposed an O($(mn + n^2)$logmn) time algorithm for

[1]A preliminary version of this paper was presented in the Second Canadian Conference on Computational Geometry, August 1990. A part of this work was done when the first two authors were with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 560012, India.

computing the feasible region for the case where M is a convex polygon. Fortune [5] improved the time bound to $\mathrm{O}(mn\log mn)$. Avnaim and Boissonnat [1] considered the case where M is a simple polygon and showed how to compute the feasible region in $\mathrm{O}(m^2n^2\log mn)$ time.

In this paper we first propose an $\mathrm{O}(mn^2)$ time algorithm for computing the feasible region for the case where M is a monotone polygon. Then we consider the general case where M is a simple polygon and propose an $\mathrm{O}(m^2n^2)$ time algorithm for computing the feasible region. Both algorithms are shown to be optimal up to a constant factor. The results in this paper have been presented in the Second Canadian Conference on Computational Geometry [9].

Earlier approaches [1,2] to computing the feasible region involve solving several subproblems on convex polygons by partitioning the *holes* in N, and the *concavities* of N (i.e. polygons outside N and inside the convex hull of N) into convex pieces. Avnaim and Boissonnat [1] follow the same approach in computing the feasible region for the general case where M is a simple polygon. They partition even M into convex pieces. In our algorithms only the holes and concavities of N are partitioned into triangles and M is left as it is. In both our algorithms we first solve subproblems involving M and triangles using the algorithm in [5]. Finally we combine the solutions to these subproblems to compute the feasible region. The optimality of our algorithm for the case where M is a monotone polygon depends crucially on an interesting intersection property of monotone polygons and the fact that M is not decomposed into convex polygons.

Now we introduce a few definitions and notations. A *simple polygon* P is represented by a sequence of distinct points in the plane $v_1, v_2, ..., v_n$, called *vertices* such that the *edges* of P, viz. v_1v_2, $v_2v_3, ..., v_{n-1}v_n, v_nv_1$, do not intersect internally. The edges of P define the boundary $bd(P)$, which divides the plane into two parts, the bounded interior $Int(P)$, and the unbounded exterior $Ext(P)$. If we traverse $bd(P)$ such that $Int(P)$ is to the right (respectively, left), then we say that $bd(P)$ is traversed in *clockwise* (respectively, *counterclockwise*) order. We assume that the vertices are numbered in counterclockwise order along the boundary of P. We also use the symbol P to denote the set of all points in $bd(P)$ and $int(P)$. A *multiply connected polygon* is a simple polygon with possibly a number of disjoint simple polygonal holes in its interior. The boundary of such a polygon is given as a set of cycles of edges: one cycle giving the counterclockwise sequence of edges of the outer boundary of the polygon and one cycle for each hole, giving the clockwise sequence of edges bounding the hole. From now onwards we refer to a multiply connected polygon as a polygon. A *polygonal region* is the union of a finite number of polygons, segments and points. The boundary of the polygonal region P is denoted by $bd(P)$. We consider the placements of a polygon P obtained by only translations. Note that a placement of P is uniquely determined by a placement of an arbitrary fixed point in P called the *reference point.* Given polygons P and Q, the set of all placements of P such that P does not intersect $Ext(Q)$ is called the *feasible region* and is denoted $I(P,Q)$. Note that $I(P,Q)$ is a closed set. The set of all placements of P such that P intersects $Int(Q)$ is denoted by $E(P,Q)$. Note that $E(P,Q)$ is an open set. $E(P,Q)$ is the same as the *configuration space* of P and Q [7,8]. Let $E'(P,Q)$ denote the complement of $E(P,Q)$. Given a polygon P, we call the exterior $Ext(P)$ as the *complement polygon* P' of P. It can be seen that $E'(P,Q') = I(P,Q)$. If M and N are polygons then it is easy to see that $I(M,N)$ is a polygonal region. Given two polygonal regions P and Q we denote their set difference by $\mathrm{P}\backslash\mathrm{Q}$. Note that $\mathrm{P}\backslash\mathrm{Q}$ is a polygonal region. A polygon P is said to be *monotone* with respect to a line l if the intersection of any line perpendicular to l with P is a connected segment, possibly empty.

This paper is organized as follows. In Section 2 we propose an optimal algorithm for computing the set of all placements by translation of a monotone polygon inside an arbitrary polygon with holes. In Section 3 we propose an optimal algorithm for computing the set of all placements by translation of a simple polygon. In Section 4 we conclude with some remarks.

2 Placement of a Monotone Polygon

In this section we present an optimal $O(mn^2)$ time algorithm for computing the feasible region $I(M,N)$, where M is a monotone polygon having m edges and N is a multiply connected polygon having a total of n edges. Let NH and MH denote the convex hulls of N and M respectively. Consider $NH \backslash N$. The polygons in $NH \backslash N$ are the concavities and holes of N. Let T denote the set of triangles obtained by triangulating the polygons in $NH \backslash N$ (Figure 1). So $N = NH \backslash (\cup_{D \in T} D)$. In order to place M so that it does not intersect $Ext(N)$ it is enough to place M inside NH and outside each D, $D \in T$. Thus $I(M,N) = I(M,NH) \backslash (\cup_{D \in T} E(M,D))$. Since NH is convex it is easy to see that $I(M,NH) = I(MH,NH)$ [3]. Thus we have the following proposition.

Proposition 2.1. $I(M,N) = I(MH,NH) \backslash (\cup_{D \in T} E(M,D))$.

Now we present an outline of the algorithm for computing $I(M,N)$.

Algorithm 2.2.

Input: A monotone polygon M (we assume without loss of generality that M is monotone with respect to the X-axis), and a multiply connected polygon N.

Output: The boundary $bd(I(M,N))$, of the set of all placements by translation of M inside N.

Step 1: Compute the convex hulls MH and NH of M and N respectively.

Step 2: Determine $I(MH,NH)$, the set of all placements by translation of MH so that MH lies inside NH.

Step 3: Compute the set T of triangles by triangulating the polygons in $NH \backslash N$.

Step 4: Determine $E(M,D)$, the set of all placements by translation of M so that M intersects $Int(D)$, for each $D \in T$ (See Figure 2).

Step 5: Compute $I(M,N) = I(MH,NH) \backslash (\cup_{D \in T} E(M,D))$.

Details of Algorithm 2.2. In Step 1, MH and NH can be computed in $O(m)$ and $O(n)$ time respectively, using the linear time algorithm to compute the convex hull of a simple polygon [10]. In Step 2, $I(MH,NH)$ can be computed in $O(m+n)$ time [3]. In Step 3, the triangulations of the polygons in $NH \backslash N$ can be computed in $O(n\log n)$ time using the algorithm in [6].

Now we show that Step 4 can be executed in $O(mn\log m)$ time. This is based on the following lemma.

Lemma 2.3. *If M is an m-sided polygon, monotone with respect to the X-axis and D is a convex polygon with the number of edges bounded by a constant k, then $bd(E(M,D))$ defines a simple polygon with $O(m)$ edges, monotone with respect to the X-axis. Further, $bd(E(M,D))$ can be computed in $O(m\log m)$ time.*

Proof. Assume that M is monotone with respect to the X-axis. We first show that $bd(E(M, D))$ defines a simple polygon monotone with respect to the X-axis. The polygon D is convex and thus monotone with respect to the X-axis. Let D_1 and D_2 be the lower and upper monotone chains of D, respectively. From the definition of $E(M, D)$ it follows that for any placement by translation of M, $bd(M)$ intersects $bd(D)$ but does not intersect $Int(D)$, if and only if the reference point r in M is on $bd(E(M, D))$ (Figure 2). Suppose we translate M in such a way around D that the reference point in M traverses the whole of $bd(E(M, D))$ in counterclockwise order, starting from the leftmost vertex of $bd(E(M, D))$. Due to the monotonicity of D and M, when M translates against D_1, it can be moved vertically upwards indefinitely without intersecting D. Similarly, when M translates against D_2, M can be moved vertically downwards indefinitely without intersecting D. Therefore $bd(E(M, D))$ defines a simple polygon, monotone with respect to the X-axis.

Now we show that $E(M, D)$ has $O(m)$ edges. Each edge of $bd(E(M, D))$ is either parallel to an edge of M or an edge of D. The polygon D has a constant number of edges and M has m edges. Therefore $bd(E(M, D))$ has $O(m)$ edges.

Now we show that $E(M, D)$ can be computed in $O(m\log m)$ time. Fortune [5], proposed an $O(cq\log cq)$ time algorithm for computing $I(C, Q)$ where C is a convex polygon with c edges and Q is an arbitrary polygon with q edges. The polygon Q may have holes. In particular, let C be the convex polygon D, with a constant number of edges, and let Q be a polygon with the whole plane as its interior except for a hole M, which is a monotone polygon. We can compute $I(D, Q)$ in $O(q\log q)$ time using Fortune's algorithm [5]. But $bd(I(D, Q)) = bd(E(D, Q')) = bd(E(D, M))$. Having constructed $bd(E(D, M))$, we construct $bd(E(M, D))$ from $bd(E(D, M))$ by reflection at the origin and an appropriate translation. Thus $E(M, D)$ can be computed in $O(m\log m)$ time. **Q.E.D.**

Since there are $O(n)$ elements in T, by Lemma 2.3, Step 4 requires $O(mn\log m)$ time.

Now we show that Step 5 can be executed in $O(mn^2)$ time. $I(M, N)$ is the region inside $I(MH, NH)$ and outside each $E(M, D)$, $D \in T$. $I(M, N)$ is a polygonal region: it is the union of a finite number of polygons, isolated segments and isolated points. We compute $bd(I(M, N))$ by determining the boundaries of each of these polygons, and each of these isolated segments and points. Note that $I(M, N)$ contains $bd(I(M, N))$. We intend to compute $bd(I(M, N))$ from $bd(I(MH, NH))$, and $bd(E(M, D))$, for all $D \in T$. Let X denote the set of polygons $\{I(MH, NH)\} \cup \{E(M, D), D \in T\}$ and S denote the set of segments that are the edges of the polygons in X. A vertex of $I(M, N)$ may be an endpoint of a segment in S or the intersection point of two segments in S. An edge of $I(M, N)$ may be a segment in S or an intersection free portion of a segment in S. So, in order to compute $bd(I(M, N))$ we compute all intersections between segments in S and the intersection free portions of segments in S. Before we state how to do these computations, we show in the following two lemmas that the number of segments $\aleph$, in S is $O(mn)$ and the number of intersections t, between segments is $O(mn^2)$.

Lemma 2.4. *Let P and Q be two polygons, monotone with respect to the same line. Let p and q be the number of edges of P and Q, respectively. There are at most $O(p + q)$ intersections between the edges of P and Q.*

Proof. Let P and Q be polygons, monotone with respect to the X-axis with p and q edges, respectively. Let P_1 and P_2 be the lower and upper monotone chains of P, respectively. Similarly, let Q_1 and Q_2 be the lower and upper monotone chains of Q, respectively. Both P_1 and P_2 may intersect Q_1 and Q_2. It is enough to show that there are at most $O(p + q)$ intersections between the edges of P_1 and Q_1. In order to count the intersections we assign intersections to the edges

of P_1 and Q_1 as follows. Let the leftmost and rightmost edges of Q_1, that intersect an edge e of P_1, be e_1 and e_k respectively. The edge e may intersect all edges e_1, e_2,..., e_k of Q_1. Assign the intersection between e_i and e to the edge e_i, $1 \leq i \leq k$. Due to the monotonicity of P_1 and Q_1, e_i, $1 < i < k$, intersects only the edge e of P_1. However, e_1 and e_k may intersect edges of of P_1 other than e. Assign these intersections to the respective edges of P_1 that intersect e_1 or e_k. Since e intersects both e_1 and e_k, e is assigned two intersections. The edges e_i, $1 \leq i \leq k$, are assigned one intersection each. In this manner all intersections are assigned to edges of P_1 and Q_1. Since each edge is assigned atmost two intersections and the total number of edges is $\mathrm{O}(p+q)$, the total number of intersections between edges in P_1 and Q_1 is $\mathrm{O}(p+q)$. **Q.E.D.**
Lemma 2.5. *The number $\aleph$ of segments in S is $O(mn)$ and the number t of intersections between the segments in S is $O(mn^2)$.*
Proof. First we show that $\aleph$=$\mathrm{O}(mn)$. $I(MH, NH)$ has $\mathrm{O}(n)$ edges [3]. From Lemma 2.3 we know that each $E(M, D), D \in T$ has $\mathrm{O}(m)$ edges. Since T has at most $\mathrm{O}(n)$ triangles, the total number $\aleph$ of line-segments in S is $\mathrm{O}(mn)$.

Now we show that t=$\mathrm{O}(mn^2)$. $I(MH, NH)$ is a convex polygon and thus monotone with respect the X-axis. We know from Lemma 2.3 that $E(M, D)$, for all $D \in T$ is monotone with respect to the X-axis. Therefore, by Lemma 2.4 the number of intersections between the edges of $I(MH, NH)$ and $E(M, D)$ is $\mathrm{O}(m+n)$. Thus the total number of intersections between the edges of $I(MH, NH)$ and $bd(E(M, D))$, for all $D \in T$ is $\mathrm{O}(mn+n^2)$. From Lemma 2.4 it follows that for $D, D' \in T, D \neq D'$, the number of intersections between the edges of $E(M, D)$ and $E(M, D')$ is $\mathrm{O}(m)$. Considering all $\mathrm{O}(n^2)$ pairs of elements in T, we find that the total number of intersections between the edges of $E(M, D)$, for all $D \in T$ is $\mathrm{O}(mn^2)$. Therefore t=$\mathrm{O}(mn^2)$. **Q.E.D.**

It is easy to compute the $O(mn^2)$ intersections as well as intersection free portions of the $O(mn)$ segments of S in two steps. First we compute pairwise intersections of the monotone polygons $E(M, D)$, $D \in T$; the intersections in each of the $O(n^2)$ pairs of monotone polygons can be computed in $O(m)$ time (see Lemma 2.4). Hence, this step requires $O(mn^2)$ time and space. Then we intersect the segments resulting from the first step with the convex polygons $I(MH, NH)$. This last step also requires $O(mn^2)$ time. After computing all the intersections as above we can easily construct a planar subdivision which is represented as a planar graph $G(V, E)$ where (1) V is the set of endpoints of all the segments in S and the intersection points of segments in S and (2) E is the set of edges $\{v_i, v_j\}$ where $v_i, v_j \in V$ and v_iv_j is an intersection free portion of a segment in S.

Once we have computed the planar subdivision G, we traverse G to mark edges of G that comprise $bd(I(M, N))$. If an edge is marked in the traversal, then the direction of traversal is assigned to the edge. These directions are so assigned that if we traverse the marked edges in the assigned directions, then $Int(I(M, N))$ lies to the left and $Ext(I(M, N))$ lies to the right (Figure 3). Note that $bd(I(M, N))$ can have several cycles of edges. Since $I(M, N)$ lies inside $I(MH, NH)$ (see Proposition 2.1), in the marking process we traverse $bd(I(MH, NH))$ keeping $Int(I(MH, NH))$ to the left and assign the direction of traversal to the edges being marked. The marking process also traverses $bd(E(M, D))$, for all $D \in T$. Since $I(M, N)$ lies outside $E(M, D)$, for all $D \in T$ (see Proposition 2.1), we traverse $bd(E(M, D))$, for each $D \in T$, keeping $Int(E(M, D))$ to the right and $Ext(E(M, D))$ to the left and assign the direction of traversal to the edges being marked.

Now we state the details of the scan of $bd(I(MH, NH))$. In the initial step, we consider any vertex v of $I(MH, NH)$ and determine the number of polygons $E(M, D)$, $D \in T$, such that v lies inside $E(M, D)$. This number is stored in a counter. Then we move edge by edge in G along

$bd(I(MH, NH))$, keeping $Int(I(MH, NH))$ to the left. Whenever we enter (respectively, leave) a polygon $E(M, D)$, $D \in T$, we increment (respectively, decrement) the counter. The edges traversed when the counter is zero are marked as edges of $bd(I(M, N))$ (Figure 3). Now we analyze the time complexity of the traversal. In the subdivision G we can move from an edge to the next edge along $bd(I(MH, NH))$ in O(1) time. Each edge is traversed at most once. Now consider the initial step that determines the membership of v in $E(M, D)$, for each $D \in T$. Since $E(M, D)$ has O(m) edges, this requires O(m) time for each $D \in T$. Since T has O(n) elements, the initial step requires O(mn) time.

Now we consider the traversal of $bd(E(M, D))$, for any $D \in T$. In the initial step, we consider any vertex v of $bd(E(M, D))$ and determine the number of polygons $E(M, D')$, $D' \in T$, $D' \neq D$, such that v lies inside $E(M, D')$. This number is stored in a counter. Then we move edge by edge in G along $bd(E(M, D))$, keeping $Int(E(M, D))$ to the right. Whenever we enter (respectively, leave) a polygon $E(M, D'))$, $D' \in T$, we increment (respectively, decrement) the counter. We also keep track whether we are inside $I(MH, NH)$. The edges of $bd(E(M, D))$ that are inside $I(MH, NH)$ and outside $E(M, D')$, for all $D' \in T$, are marked as edges of $bd(I(M, N))$ (Figure 3). Now we analyze the time complexity of the traversal. In the subdivision G we can move from an edge to the next edge along $bd(I(MH, NH))$ in O(1) time. Each edge is traversed at most once. Now consider the initial step that determines the membership of v in $E(M, D')$, for all $D' \in T, D' \neq D$. Since $E(M, D')$ has O(m) edges, this requires O(m) time for each D'. Since T has O(n) elements, the initial step requires O(mn) time.

We repeat the above mentioned traversal for each $E(M, D)$, $D \in T$. The size of G is O(mn^2) and each edge is traversed at most once. So the total time required is O(mn^2). This completes the traversal of G and the marking of edges.

Once the markings are done we scan the marked edges to find the isolated segments of $I(M, N)$. The edges that have been marked in both directions are isolated segments. The remaining edges form cycles enclosing unconnected regions of $I(M, N)$. In Figure 3 there are seven such regions and two isolated segments in $I(M, N)$. This completes the computation of $I(M, N)$ in Step 5.

We summarize our result in the following theorem.

Theorem 2.6. *Let M be a monotone polygon with m edges and N be a multiply connected polygon with n edges. $I(M, N)$, the set of all placements of M by translation inside N, can be computed in $O(mn^2)$ time.*

In Figure 4, M is a monotone polygon and $I(M, N)$ has $\Omega(mn^2)$ vertices. So our algorithm is asymptotically optimal.

3 Placement of a Simple Polygon

In this section we present an O(m^2n^2) time algorithm for computing $I(M, N)$ where M is a simple polygon with m edges and N is a multiply connected polygon with n edges. The main steps of the algorithm are the same as those of Algorithm 2.2 of the previous section. It is sufficient to consider Steps 4 and 5. First we show that Step 4 can be computed in O($mn\log m$) time using the following lemma.

Lemma 3.1. *If M is an m-sided simple polygon and D is a convex polygon with the number of edges bounded by a constant k, then $bd(E(M, D))$ defines a multiply connected polygon with $O(m)$ cycles of edges and a total of $O(m)$ edges. Further, $bd(E(M, D))$ can be computed in $O(m\log m)$ time.*

Proof: The proof that $bd(E(M,D))$ has $\mathrm{O}(m)$ edges and that $bd(E(M,D))$ can be computed in $\mathrm{O}(m\log m)$ time follows from [5] along the lines similar to proof of Lemma 2.3. From [5] it is clear that $E(M,D)$ defines a multiply connected polygon with atmost $\mathrm{O}(m)$ cycles of edges. **Q.E.D.**

Since T has $\mathrm{O}(n)$ elements, computing $E(M,D)$ for all $D \in T$ requires $\mathrm{O}(mn\log m)$ time. This completes the analysis of the time complexity of Step 4.

Now consider Step 5. As in Section 2, let X denote the set of polygons $\{I(MH,NH)\} \cup \{E(M,D), D \in T\}$ and S denote the set of segments that are the edges of the polygons in X. Consider the planar subdivision G, induced by the segments in S as defined in Section 2. Here, we compute the planar subdivision G using the algorithm in [4]. Since M is a simple polygon, $bd(E(M,D))$, $D \in T$ is an m-sided multiply connected polygon (Lemma 3.1). Observe that $bd(E,D))$ and $bd(E(M,D'))$, $(D,D' \in T, D \neq D')$, have $\mathrm{O}(m^2)$ intersections. Since T has $\mathrm{O}(n)$ elements, there are $\mathrm{O}(n^2)$ pairs (D,D'). So, G has $\mathrm{O}(m^2n^2)$ vertices and edges. As in Section 2, Step 5 involves a scan of $bd(I(MH,NH))$, and $bd(E(M,D))$ for each $D \in T$. The scan of $bd(I(MH,NH))$ is similar to that in the previous section. So we consider only the scan of $bd(E(M,D))$, $D \in T$. Observe that $E(M,D)$ is a multiply connected polygon and hence $bd(E(M,D))$ has $\mathrm{O}(m)$ cycles of edges (Lemma 3.1). We consider each cycle of $bd(E(M,D))$ separately. Let v be a vertex of G on a cycle C of $bd(E(M,D))$. We consider the time complexity of only the initial step for cycle C. The analysis of the time complexity of the remaining steps is similar to that in the previous section. In the initial step we determine the number of polygons $E(M,D')$, $D' \in T$, $D' \neq D$, such that v lies inside $E(M,D')$. This is done by determining whether v lies inside the region enclosed by a cycle of $bd(E(M,D'))$, for each $D' \in T$. Since there are $\mathrm{O}(n)$ elements in T and each $E(M,D')$, $D' \in T$ has $\mathrm{O}(m)$ edges, the initial step for the cycle C requires $\mathrm{O}(mn)$ steps. Since there are $\mathrm{O}(m)$ cycles in $bd(E(M,D))$, the total time required for the initial step for $bd(E(M,D))$ is $\mathrm{O}(m^2n)$. Repeating the initial step for $bd(E(M,D))$, $D \in T$ for all the $\mathrm{O}(n)$ elements D of T, requires $\mathrm{O}(m^2n^2)$ time. Thus Step 5 requires $\mathrm{O}(m^2n^2)$ time.

We summarize our result in the following theorem.

Theorem 3.2. *Let M be a simple polygon with m edges and N be a multiply connected polygon of n edges. $I(M,N)$, the set of all placements of M by translation inside N can be computed in $O(m^2n^2)$ time.*

In Figure 5 it is evident that $I(M,N)$ has $\Omega(m^2n^2)$ vertices. So our algorithm is optimal up to a constant factor.

4 Conclusion

Let M be an m-sided polygon, monotone with respect to two mutually perpendicular lines. We call such polygons *rectilinearly convex*. Let N be an n-sided polygon with holes. Since M is a monotone polygon we can compute the feasible region using the algorithm in Section 2 in $\mathrm{O}(mn^2)$ time. However, the best known lower bound on the size of the feasible region is $\Omega(mn+n^2)$ (Figure 6). So, designing an optimal algorithm for computing the feasible region for a rectilinearly convex polygon remains an open question. Another future direction is to consider the placement of other classes of polygons viz., star, spiral and unimodal polygons.

REFERENCES

1. F. Avnaim, J.-D. Boissonnat, Simultaneous containment of several

polygons, *Proc. 3rd ACM Symposium on Computational Geometry*, Waterloo, Ontario, Canada (1987) 242-250.

2. B.S. Baker, S.J. Fortune, S.R. Mahaney, Polygon containment under translation, *J. of Algorithms*, 7(4) (1986) 532-548.

3. B. Chazelle, The polygon containment problem, *Advances in Computing Research, Vol. 1, JAI Press*, (1983) 1-33.

4. B. Chazelle, H. Edelsbrunner, An optimal algorithm for intersecting line segments in the plane, *Proc. 29th Annual Symposium on Foundations of Computer Science*, (1988).

5. S. J. Fortune, A fast algorithm for polygon containment by translation, in: W. Brauer ed. *Proc. 12th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science, Vol. 194*, (Springer Verlag, 1985) 189-198.

6. M. R. Garey, D. S. Johnson, F. P. Preparata, R. E. Tarjan, Triangulating a simple polygon, *Information Processing Letters*, 7(4) (1978) 175-179.

7. T. Lozano-Perez, Spatial planning: A configuration space approach, *IEEE Transactions on Computers*, 32(2) (1983) 108-120.

8. T. Lozano-Perez, M. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, *Comm. of the ACM*, 22(10) (1979) 560-570.

9. S.P. Pal, B. Dasgupta, C.E. Veni Madhavan, Optimal polygon placement by translation, *Proc. 2nd Canadian Conference on Computational Geometry*, Ottawa, Ontario, Canada (1990) 164-171.

10. F.P. Preparata, M.I. Shamos, *Computational Geometry-An Introduction*, Springer-Verlag, 1985.

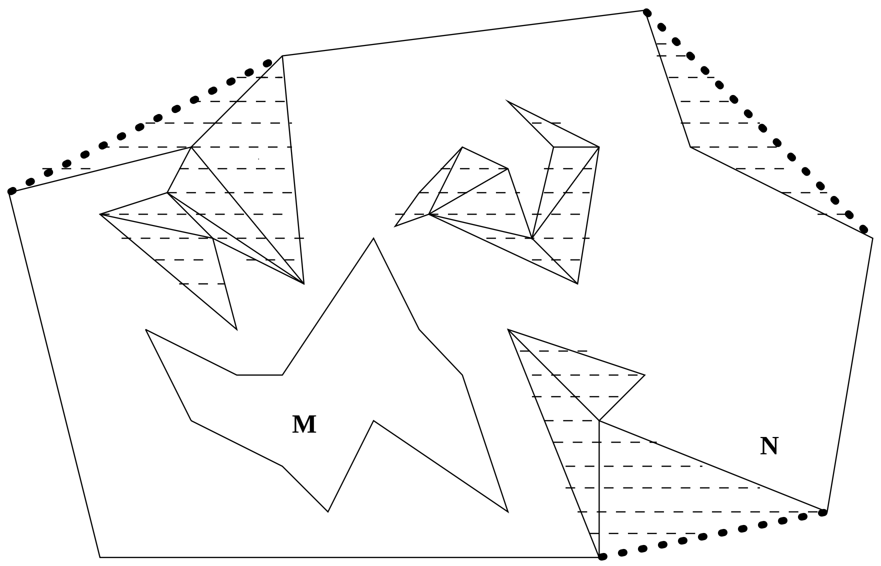

Figure 1. Triangulation of polygons in NH-N

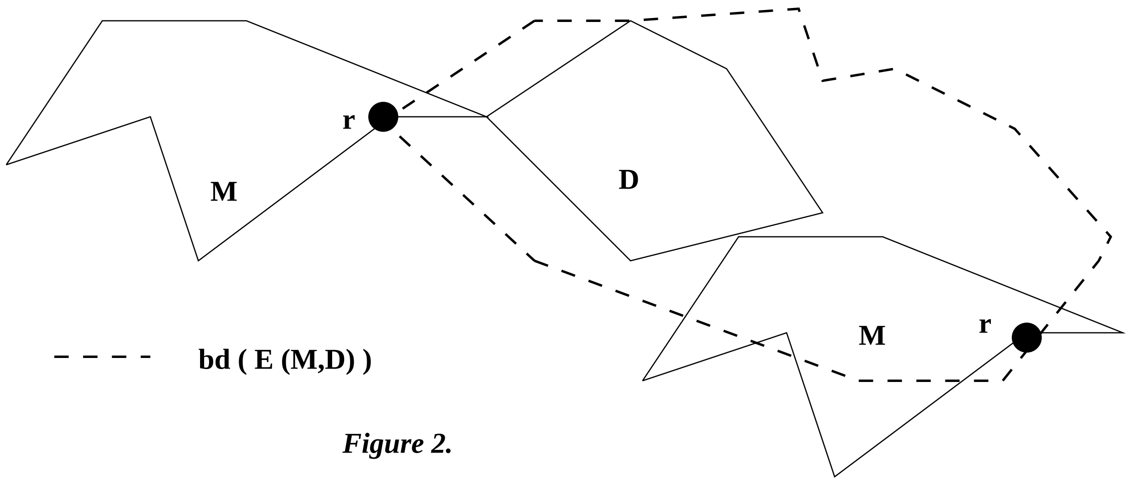

Figure 2.

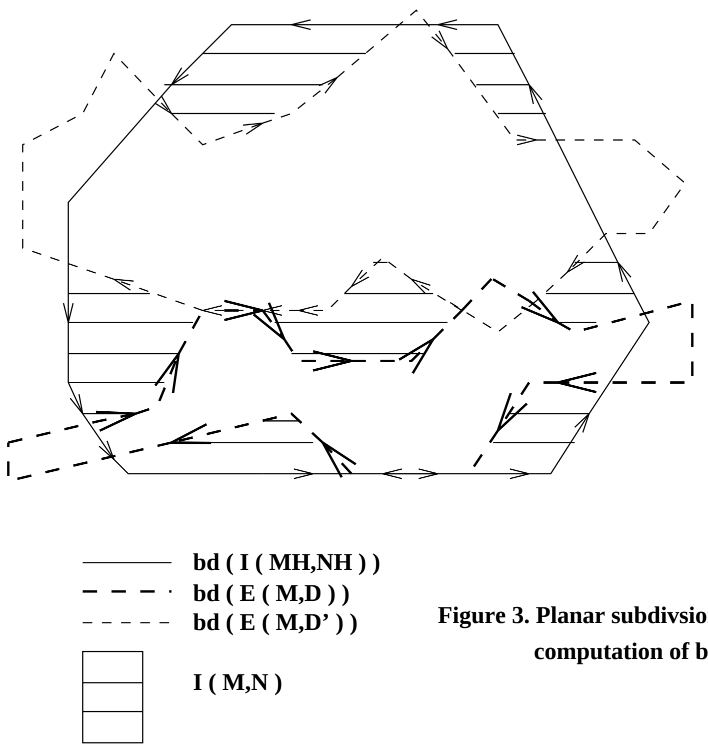

Figure 3. Planar subdivsion G and computation of bd(I(M,N))

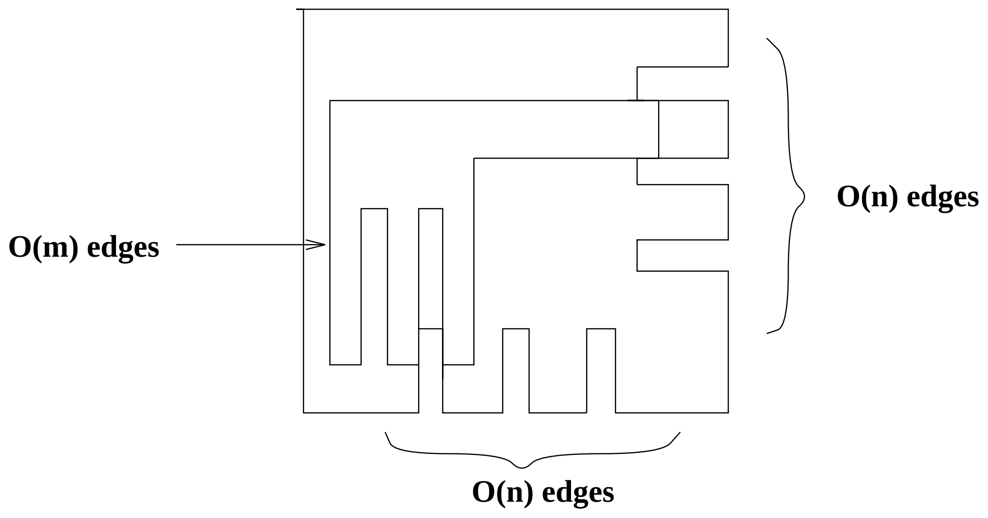

Figure 4. I(M,N) has Ω (m,n ^{2}) vertices

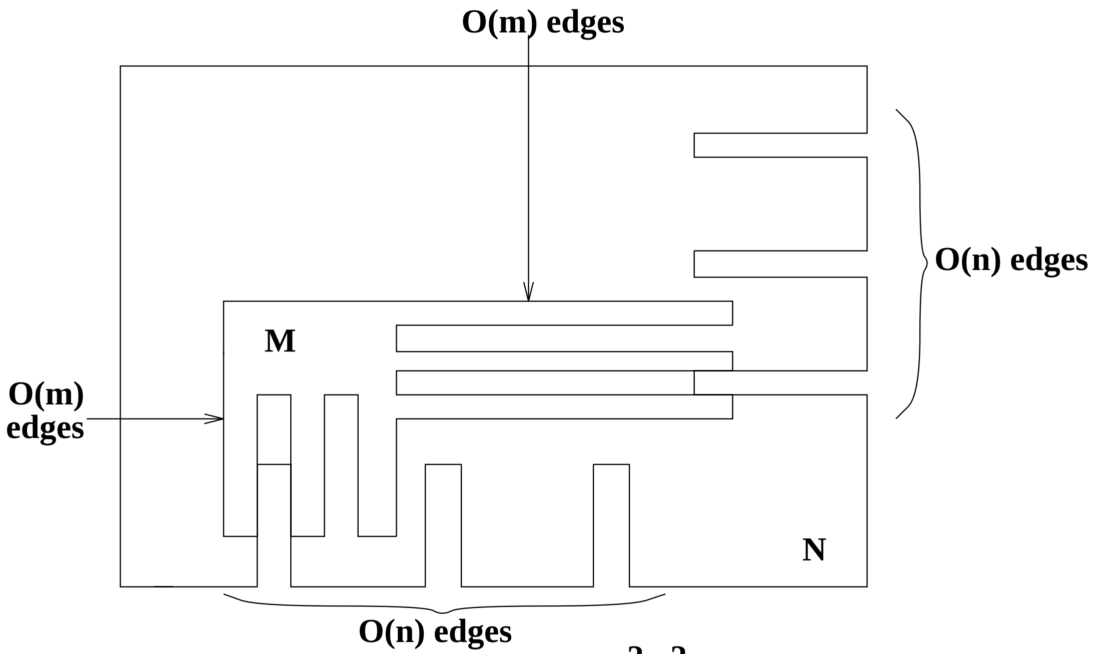

Figure 5. I(M,N) has $\Omega(m^2 n^2)$ vertices.

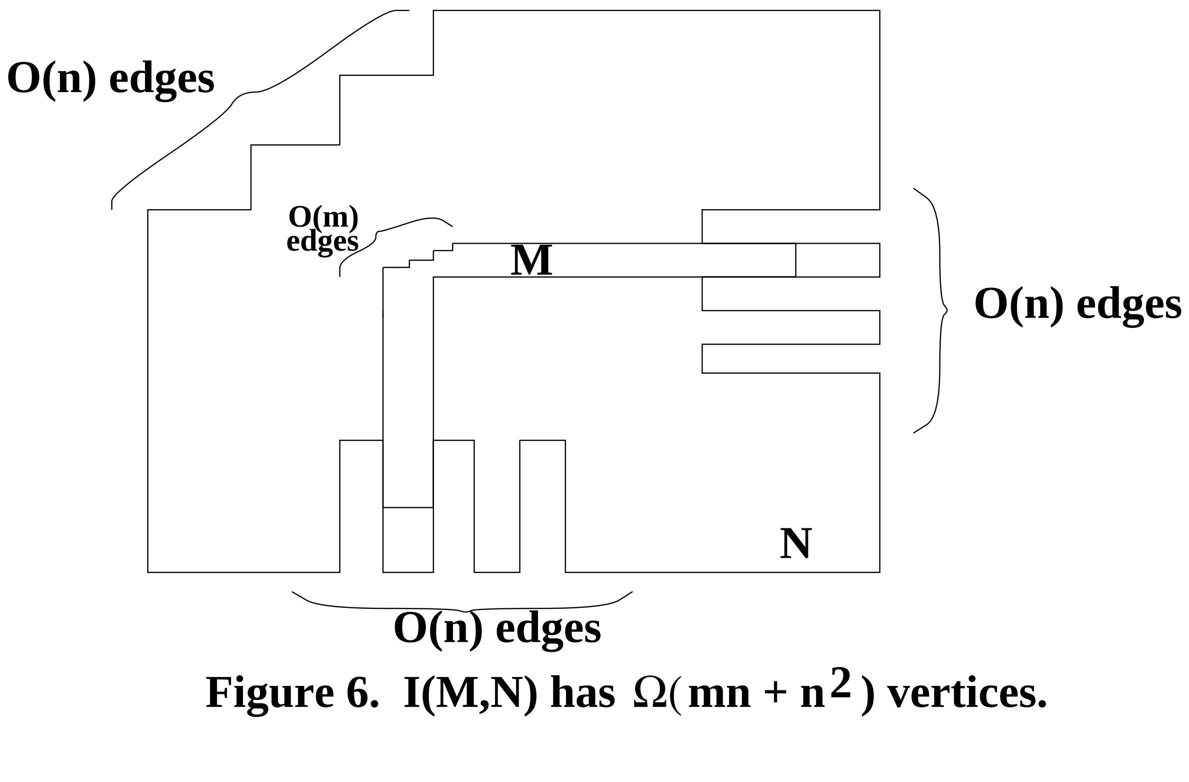

Figure 6. I(M,N) has $\Omega(mn + n^2)$ vertices.