Optimal polygon placement by translation

Sudebkumar Prasant Pal * Bhaskar Dasgupta† C. E. Veni Madhavan ‡

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 within N without intersecting any hole. First we propose an $O(mn^2)$ 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 $O(m^2n^2)$ time algorithm for computing the feasible region. Both algorithms are optimal up to a constant factor.

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

In this paper we first propose an $O(mn^2)$ 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 $O(m^2n^2)$ time algorithm for computing the feasible region. Both algorithms are shown to be optimal up to a constant factor.

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 using the optimal algorithm for computing intersections of segments in the plane [4]. The optimality of our algorithm for the

*Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, INDIA, email: uunet!shakti!vigyan!cevm

†Computer Science Department, The Pennsylvania State University, 333 Whitmore Laboratory, University Park, PA 16802, U.S.A., email: bhaskar@psuvax1.cs.psu.edu

‡Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, INDIA, email: uunet!shakti!vigyan!cevm

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 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. 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. 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 $NH-N$. The polygons in $NH-N$ are the concavities and holes of N. Let T denote the set of triangles obtained by triangulating the polygons in $NH-N$ (Figure 1). So $N = NH - (\cup_{D\in T} D)$. 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 $I(M,N) = I(M,NH) - (\cup_{D\in T} E(M,D))$. 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. $I(M,N) = I(MH,NH) - (\cup_{D\in T} E(M,D))$.

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 $NH - N$.

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

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 $NH - N$ 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 which we state without proof.

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.*

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.*

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 $O(mn^2)$.* Therefore t=$O(mn^2)$. **Q.E.D.**

We compute intersections as well as intersection free portions of segments in S using the algorithm of Chazelle and Edelsbrunner [4]. Their algorithm computes the planar subdivision induced by the segments in S and runs in $O(\aleph \log \aleph + t)$ time i.e. $O(mn^2)$ time (See Lemma 2.5). The subdivision 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 $v_i v_j$ 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. We omit the details of the procedure that traverses the subdivision G to mark the edges of $bd(I(M, N))$. For details see [9]. The edges that get marked in both directions are isolated segments of $bd(I(M, N))$. The remaining edges form cycles, each enclosing a connected region of $I(M, N)$. In Figure 3

there are seven such regions and two isolated segments in $I(M,N)$. All computations required in determining $I(M,N)$ from the subdivision G require a total of $O(mn^2)$ time. 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 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. Using arguments similar to those in Section 2, it can be shown that Step 4 can be computed in $O(mn\log mn)$ time and Step 5 can be computed in $O(m^2n^2)$ time. We omit the details of these steps in this extended abstract. For details see [9].

We summarize our result in the following theorem.

Theorem 3.1. *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 $O(mn^2)$ 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. of the Third ACM Symposium on Computational Geometry*, pp. 242-250, 1987.

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

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

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

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

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

7. T. Lozano-Perez, Spatial planning: A configuration space approach, *IEEE Transactions on Computers, Vol. C-32, Feb. 1983*, pp. 108-120.

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

9. S.P. Pal, B. Dasgupta, C.E. Veni Madhavan, Optimal polygon placement by translation, Technical Report No. IISc-CSA-90-7, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 560012, INDIA.

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

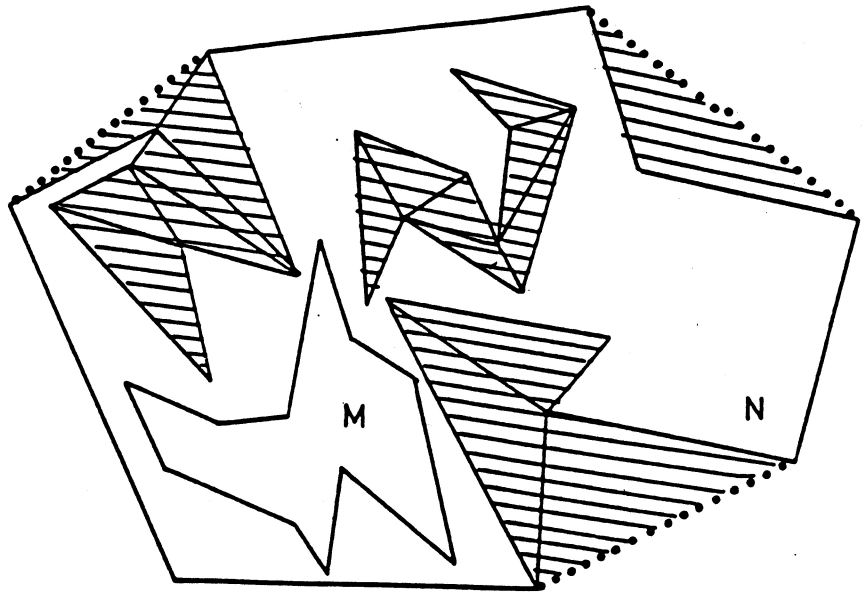
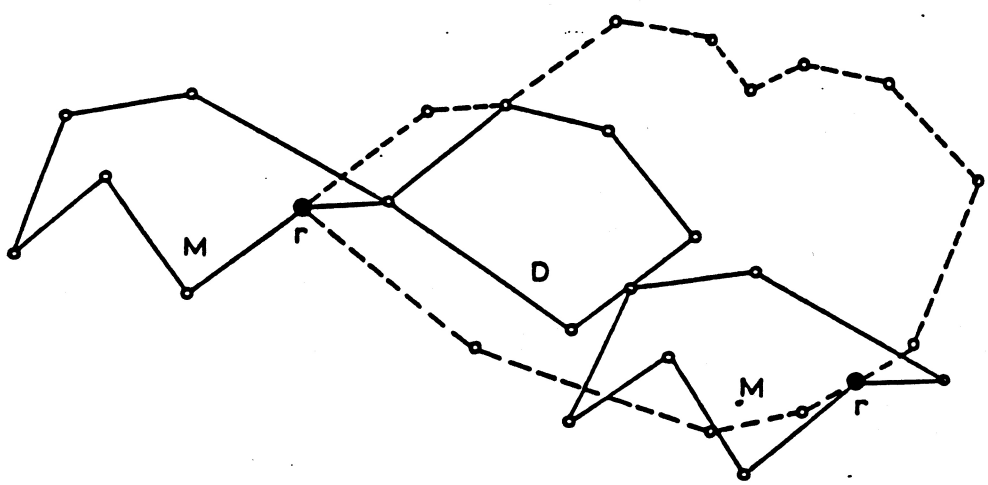


Figure 1 . Triangulation of polygons in NH−N



----- bd (E (M, D))

Figure 2

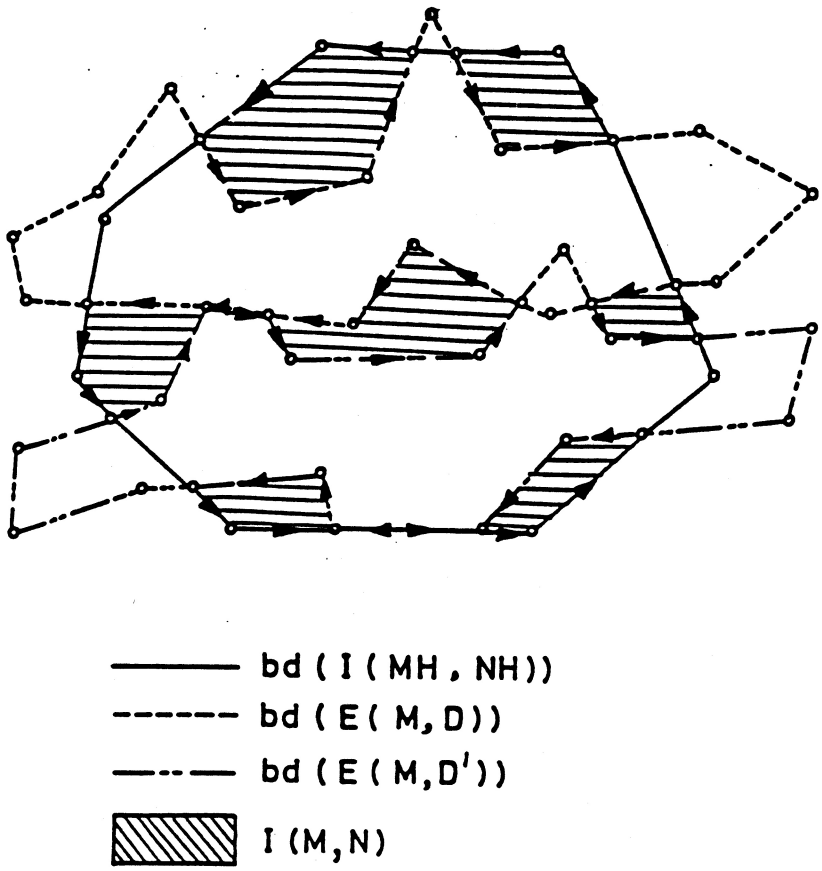


Figure 3 . Planar subdivision G and computation of bd(I (M,N))

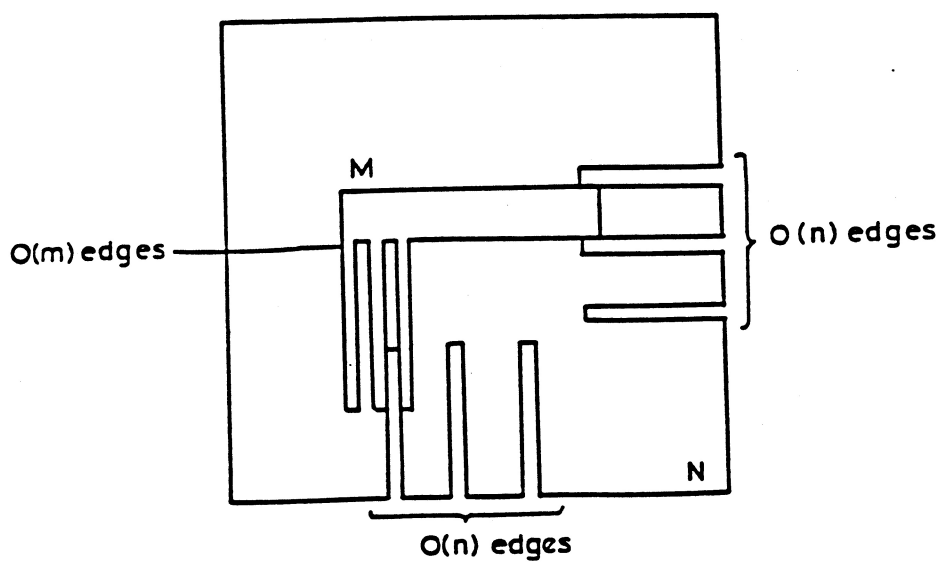


Figure 4 I (M ,N) has Ω (mn^2) vertices

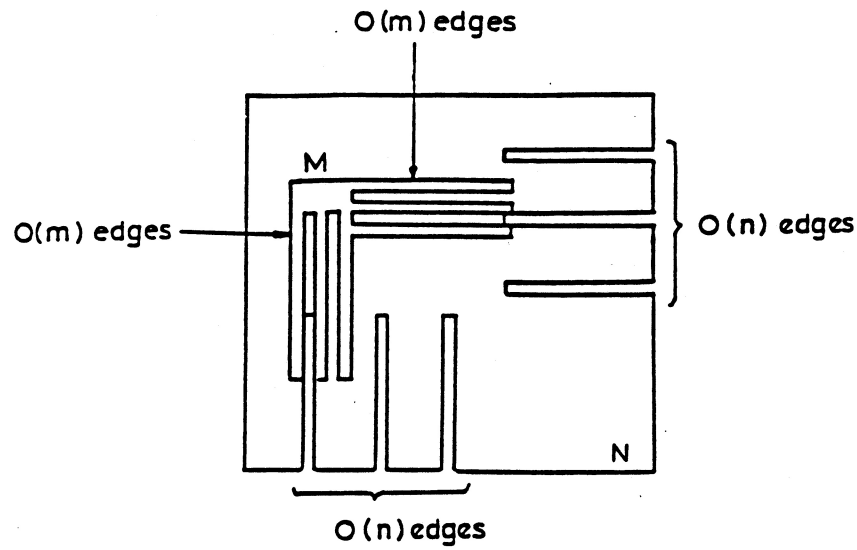


Figure 5 $I(M,N)$ has $\Omega(m^2n^2)$ vertices

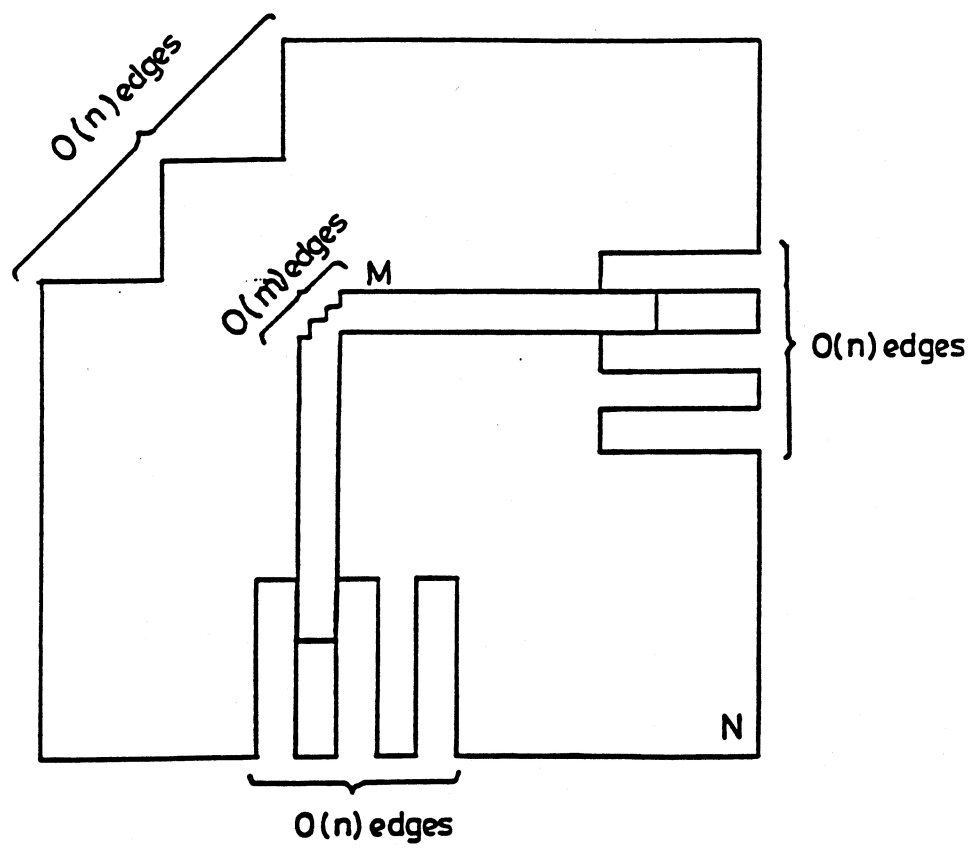


Figure 6 $I(M,N)$ has $\Omega(mn+n^2)$ vertices