設計自動化　40－5
(1987. 12. 17)

Generalized Manhattan Path Algorithm with Applications

浅野　哲夫

Tetsuo Asano

大阪電気通信大学工学部応用電子工学科

Osaka Electro-Communication University

1. Introduction

In this paper we first consider the problem known as Manhattan Path problem stated as follows. We are given a set S of orthogonal line segments (horizontal and vertical). When two line segments s and t in S are specified as source and target segments, we want to find an alternative sequence of horizontal and vertical line segments connecting s with t only using line segments of S. This problem was first proposed and solved by Lipski [Li83, Li84]. He presented an efficient algorithm for this problem which runs in O(n log n) time using O(n log n) space, where n is the number of given line segments. Note that the time and space does not depends on the number of intersections between line segments.

In Section 2 we consider a generalization of the above Manhattan path problem. In the new problem we are given a set R of connected regions in addition to a set S of line segments. Then we place a condition that every intersection of successive line segments on a route to connect the source and target line segments must lie on some connected region of R. Thus the Manhattan path problem can be considered as a special case of this problem in which R is the entire plane. For this generalized problem we present an algorithm which runs in O(n log n) time and O(n) space or $O(n \log^2 n)$ time and space depending on the data structure used. This algorithm always finds a route between two specified line segments if any like Lipski's algorithm.

In Section 3, based on the proposed algorithm we solve another practical problem which is connected with a layout design of bipolar LSIs. The purpose is to find an orthogonal wiring route of predetermined width between pairs of terminals avoiding polygonal obstacles in two layers. The route must consist only of horizontal and vertical wire segments and wire segments on different layers must be connected by large rectangular vias. The most difficult is to decide at any specified point whether there is a room for a via to be placed. Also, each terminal may be of an arbitrary rectilinear polygonal shape, and a wiring route may start and end at any part of a terminal. The width of wiring segments may be specified arbitrarily for each layer. Two layers are available for interconnection; one layer called the X-layer is used only for horizontal wiring routes and the other called the Y-layer used only for vertical ones. Horizontal wire egments have width $2*w_y$ and vertical wire segments have width $2*w_x$. Horizontal wire segments are connected with vertical wire segments by vias of size $2*d_x \times 2*d_y$. We assume terminals occupy the X-layer only while vias occupy both layers. Then, the problem of finding a route between two specified terminals can be formulated as an instance of the generalized Manhattan path problem described above.

2. Generalized Manhattan Path Algorithm

We first describe the Manhattan Path Problem.

[Manhattan Path Problem] Given a set of horizontal and vertical line segments and specified two line segments s and t, find a path between s and t using given line segments.

The problem was first proposed by Lipski and solved by himself [Li83,Li84]. His algorithm runs in O(n log n) time with O(n log n) space even in the worst case where n is the number of line segments. The key idea is that the problem can be solved by a breadth-first-search, which can be executed by at most n DELETE operations, which delete line segments from a set, and n LIST operations which enumerate all the line

segments intersecting a query line segment. The data structure used is a combination of segment tree [Be79] and the Gabow and Tarjan's linear time version of the union-find algorithm on consecutive sets. We could also use a similar data structure proposed by Imai and Asano [IA84]. We could implement Lipski's algorithm in linear space by using priority search tree [Mc85], a data structure which was originally devised to deal with a set of points in the plane, although it needs O($n \log^2 n$) time instead. In [Mc85, AS086], it is shown that LIST can be done in O($\log^2 n + k$) time where k is the number of line segments to be enumerated in a dynamic environment in which DELETE of a line segment is performed in O(log n) time. The priority search tree requires only linear storage. Note that in either case the time and space does not depend on the number k of intersections between given line segments. Of course, we could implement the Lipski's algorithm without DELETE operations while in this case O(n log n + k) time may be needed in the worst case.

We consider a generalization of the above Manhattan path problem. In the new problem we are given a set R of connected regions in addition to a set S of line segments. Then we place a condition that every intersection of successive line segments on a route to connect the source and target line segments must lie on some connected region of R. Thus the Manhattan path problem can be considered as a special case of this problem in which R is the entire plane. For this generalized problem we present an algorithm which runs in O(n log n) time and O(n log n) space or O($n \log^2 n$) time and O(n) space. This algorithm always finds a route between two specified line segments if any like Lipski's algorithms.

[Generalized Manhattan Path Problem] We are given a set S of orthogonal (horizontal and vertical) line segments and a set R of connected regions bounded by orthogonal line segments, which are assumed to be parts of those of S. When two line segments s and t are specified, find an alternating sequence of horizontal and vertical line segments of S such that each intersection between two successive line segments lies in the interior of some region of R.

An example is given in Fig. 1, which consists of 20 line segments (10 horizontal and 10 vertical) and 5 connected regions. Note that the region R1 is not simple in the strict sense. Some regions may contain holes and other may be contained in a hole of some other connected region. On eof the solutions to the problem given in Fig. 1 is (s=h0, v3, h3, v6, h6, v8=t). The sequence (s=h0, v3, h3, v6, h9, v8=t) is not a solution since the intersection between h9 and v8 is not contained in any connected region.

One way to solve this problem is to combine the Lipski's algorithm with point location technique (for example, see [Ki83], [Li84], and [EKA84]), which finds the name of a region containing an arbitrary specified point in O(log n) time with O(n log n)-time preprocessing, where n is the number of vertices of given connected regions. In our case all the intersections are not available. In the worst case we have to examine every intersection and at each intersection we need O(log n) time for each point location to decide whether the intersection is available, that is, whether it lies in the interior of some connected region of R. Thus it leads to an O(K log n)-time algorithm where K is the number of intersections of given line segments.

In this paper we present an efficient algorithm whose time complexity does not depends on K, the number of intersections. Recall that we have assumed that every boundary portion of given connected regions is contained in some line segment. Thus, we have the following key observation.

[Lemma 1] Let si and sj be any two line segments of S intersecting the same connected region Rk. Then, there is a sequence of line segments connecting si and sj such that each intersection between two successive ones lies in the interior of Rk.

This leads to the following algorithm.

[Algorithm for Generalized Manhattan Path Problem]

```
(input)
  .s and t: source and target line segments;
  .H: a set of all horizontal line segments;
  .V: a set of all vertical line segments;
  .R: a set of connected regions;
  (* for each boundary edge e of R the line segment
     containinge is denoted by s(e) *)
(initialization)
  .for each line segment L do NEXTR[L] = NULL;
  .for each connected region Ri do NEXT[Ri] = NULL;
  .initialize a (first-in-first-out) queue QR;
   (* QR is now empty *)
  .construct data structure T(H), T(V), and T(R) which
     support LIST and DELETE operations for the sets H
     and V and the set of boundary edges of R;
   (* segment-tree type data structure or priority
     search tree *)
  .mark all the connected regions of R, that is,
     mark[Ri] = ON for each region Ri of R;
(Breadth-first search){
  .for every connected region Ri intersecting t do{
     .unmark the region Ri;
```

```
      .NEXTL[Ri] = t;
      .put Ri into QR;}
   .if t is horizontal then DELETE it from H
          else DELETE it from V;
   .while( QR is not empty and s is not deleted yet){
      .take a connected region Ri out of QR;
      .initialize a queue QL;
     (* QL will contain a set of line segments
       intersecting the region Ri *)
     .for each boundary edge e of Ri do{
        .put the line segment s(e) into QL;
        .DELETE s(e) from T(H) or T(V), depending on
          whether it is horizontal or vertical;
        .for each line segment L intersecting e do {
          .put L into QL;
          .DELETE L from T(H) or T(V);}
     }
     .while(QL is not empty) do {
        .take a line segment L out of QL;
        .NEXTR[L] = Ri;
        .enumerate all the marked connected regions
         that intersect L by performing LIST operation
         against a set of existing boundary edges,
         i.e., T(R);
        .for each such region Rj do{
          .unmark Rj;
          .NEXTL[Rj] = L;
          .put Rj into QR;}
     }
   }
   .if s has been deleted then {
      .find the alternating sequence of line segments
       and connected regions connecting s to t by
       tracing the links NEXTR[] and NEXTL[];
      .for each three tuple (Li,Rj,Lk) in the sequence
        .find a sequence of line segments connecting
         Li to Lj within the connected region Rj;
         (* see EQUENCE CONVERSION below for detail *)
   }
   else report( "no solution!" );
}
procedure unmark(Ri){
  (* Ri is a connected region *)
   .mark[Ri] = OFF;
   .DELETE all the boundary edges of Ri from T(R);}
```

In the Lipski's algorithm breadth-first search was performed with respect to line segments and a route connecting two specified line segments using the minimum number of line segments is obtained. In the above algorithm we look for a shortest possible sequence of line segments and connected regions connecting two specified line segments by performing the breadth-first search with respect to connected regions. As a result we obtain a tree as shown in Fig. 2 which has two vertex sets, one corresponding to a set of line segments and the other to a set of connected regions. The root corresponds to t, the target. It is easily seen that the above algorithm finds a route between two specified line segments which is optimal in the sense that it goes through the least connected regions.

We have seen that the algorithm finds a sequence of line segments and connected regions (s=L0, R0, L1, R1, ... , Rm, Lm = t). The only remaining problem is how to connect two line segments Li and s_{i+1} within the region Ri for each i=0, 1, ... , m. Such a sequence is easily found since the boundary of the region Ri gives one such sequence. Or we can find such a sequence connecting L_i to L_{i+1} within Ri as follows.

```
[SEQUENCE CONVERSION]
(input) a sequence (s = L0, R0, L1, R1, ... , Lm = t);
(output) a route connecting s to t: (s=L0, L1', L2', .
   .. , Lm' = t);
(method)
{ .for every line segment L do
     NEXT[L] = NULL;
  .for each i=m-1, m-2, ... , 0 do {
     .initialize a queue Q;
     .if Li+1 is contained in H or V then DELETE it;
     .put Li+1 into Q;
     .while(Q is not empty and Li has not been
       deleted) do {
       .take a line segment L out of Q;
       .compute the portion L[i] occupied by Ri;
       .for each line segment L' intersecting L[i] do{
          .DELETE L' from H or V;
          .NEXT[L'] = L;
          .put L' into Q;}
     }
  }
  .finally trace the link NEXT[] to get a sequence;
}
```

[Theorem 1] The above algorithm for the generalized Manhattan path problem together with the procedure SEQUENCE CONVERSION finds a route connecting two arbitrarily specified segments if any in $O(n \log n)$ time using $O(n \log n)$ space or in $O(n \log^2 n)$ time using $O(n)$ space.

3. Application to Layout Design

In this section as an application of the algorithm for the generalized Manhattan path problem we consider

a wire-routing problem defined as follows. We are given terminals, pads, and wire segments which have already been routed. The problem is to find an orthogonal wiring route between two specified terminals avoiding obstacles (other terminals, pads, and wiring segments already routed) under the following condition. Terminals and pads may be of arbitrary polygonal shapes, which must be rectilinear, and the route desired may start from any part of a source terminal and end at any part of target terminal. Two layers are available for interconnection; one layer called the X-layer is used only for horizontal wiring routes and the other called the Y-layer used only for vertical ones. We can specify the horizontal and vertical widths $2w_y$ and $2w_x$ of the route together with the source and target terminals. Horizontal wire segments are connected with vertical wire segments by vias of size $2d_x \times 2d_y$. We assume that terminals occupy the X-layer only while vias occupy both layer. (This assumption is based on the present technology of a bipolar LSI.) The purpose of this section is to formulate the above-stated problem as an instance of the generalized Manhattan path problem considered in the previous section and present an efficient algorithm for the problem which runs in $O(n \log n)$ time using $O(n \log n)$ space or $O(n \log^2 n)$ time using linear space, where n is the total number of vertices in a given layout (including terminals, pads, wire segments, and vias.

Initially we are given a set of terminals which become obstacles for the first net. Wire segments and vias for already routed nets become obstacles for the next net. Since wire segments and vias are of rectangular shapes, a set of obstacles can be represented by two sets of rectilinear polygons for the two layers. Let X = {P[x1], P[x2], ... , P[xm]} and Y = {P[y1], P[y2], ... , P[yn]} be the sets of obstacle polygons for the X- and Y-layers, respectively.

We assume every net is a two-terminal net. Then, we can specify a net by a pair of terminals. See, for example, Fig. 3, where the two terminals S and T are to be connected. The problem is to find a path as shown in Fig. 4. Note that horizontal (vertical, resp.) wire segments must have width $2w_y$ ($2w_x$, respectively) and vias of size $2d_x \times 2d_y$ must be placed at every intersection of horizontal andvertical wire segments. Of course, any part of the path must not have proper intersection with obstacle polygons.

In order to find such a path between two specified terminals, we find a region which can accomodate center lines of horizontal wire segments for the X-layer. Such a region, called Horizontal Routable Region denoted by HR(S, T, X, w_y), can be obtained by vertically expanding every obstacle polygon of X by w_y in the X-layer excepting S and T [OS84] (see Fig. 5). Although terminals S and T themselves are not obstacles, those horizontal wire segments which intersect them but do not have enough contact with them are useless. Thus, we shrink those terminals by w_y in the vertical direction. Then, any horizontal line segment intersecting the shrunk terminal can be a center line of a feasible horizontal wire segment which has sufficient contact with the terminal.

It is easily observed that if a horizontal line segment L is contained in Horizontal Routable Region then we can place the horizontal wire segment ws(L) of width $2w_y$ with L as its center line avoiding obstacles and that if L touches a shrunk polygon Sy(S, w_y) then the wire segment ws(L) has sufficient contact with the terminal S.

In a similar way we define Vertical Routable Region VR(S,T,Y,w_x) (see Fig. 6).

In addition to these, we deine a region called Via Acceptable Region where the center of a via can be laid without any intersection with obstacles (see Fig. 7). Since the size of a via is $2d_x \times 2d_y$, we can define the region by enlarging every obstacle polygons by specified widths.

Using these three regions, Horizontal and Vertical Routable Region HR and VR and Via Acceptable Region VA defined above, we define two sets of line segments, denoted by H and V, which are candidate for the center lines of horizontal and vertical wire segments, respectively, as follows.

H = { horizontal line segment L;
(1) L is contained in Horizontal Routable Region, and (2) L touches some horizontal boundary segment of Horizontal Routable Region or Via Acceptable Region}
and

V = { vertical line segment L;
(1) L is contained in Vertical Routable Region, and (2) L touches some vertical boundary segment of Vertical Routable Region or Via Acceptable Region}.

Such line segments can be enumerated by a plane sweep technique after resizing obstacle polygons (see Fig. 8).

Now the problem is formulated as follows.

[Two Layer Routing Problem with large Vias]

We are given two sets H and V of horizontal and vertical line segments, respectively, and a rectilinear polygonal region VA referred to as Via Acceptable Region. When two subsets Ns and Nt of H V are specified as source and target line segments, find such a path between some line segment of Ns and some of Nt

only using given line segments that each intersection between any two consecutive line segments lie in the region VA. It is easily seen that this is an instance of the generalized Manhattan path problem considered in the previous section. The via acceptable region plays a part of the connected region in the previous section. Note that boundary edges of the region are contained in some line segments of H or V, which is easily recognized from the above definition of the sets H and V. The only difference is that in this case source and target are sets of line segments. But it is easy to modify the previous algorithm so as to allow sets of line segments as source and target. Thus, we can apply the algorithm given in the previous section to solve the above two-layer routing problem.

Acknowledgment The author would like to express his gratitude for Dr. Kenji Yoshida and Dr. Mituhiro Koike of Toshiba Corporation for giving the problem with some examples and for their discussions.

References

[Be79] J.L. Bentley: "Decomposable Searching Problem," Information Processing Letters, vol.8, pp.244-251, 1979.

[Ed80] H. Edelsbrunner: "Dynamic Data Structure for Orthogonal Intersection Queries," Report 59, Institut fur Informationsverarbeitung Technishe, Universitet Graz, 1980.

[EKA84] M. Edahiro, I. Kokubo, and Ta. Asano: "A New Point Location Algorithm and Its Practical Efficiency -- Comparison with Existing Algorithms," ACM Trans. on Graphics, 3, 2, pp.89-109, 1984.

[FSO84] Y. Furuya, M. Sato, and T. Ohtsuki: "An Algorithm for One-Dimensionally Shortest Two-Layer Interconnection," (in Japanese) in Proc. of Information Processing Society of Japan, 1984.

[IA84] H. Imai and Ta. Asano: "Dynamic Segment Intersection Search with Applications, " Proc. 25th IEEE Symp. on Foundations of Computer Science, Florida, pp.393-402, 1984.

[Ki83] D.G. Kirkpatrick: "Optimal Search in Planar Subdivisions," SIAM J. Comput., 12, 1, pp.28-35, 1983.

[Li83] W. Lipski: "Finding a Manhattan path and related problems," Networks, 13, pp.399-409, 1983.

[Li84] W. Lipski: "An O(n log n) Manhattan Path Algorithm," Information Processing Letters, 19, pp. 99-102, 1984.

[LP84] D.T. Lee and F.P. Preparata: "Computational Geometry -- A Survey," IEEE Trans. on Computers, C-33, pp.1072-1101, 1984.

[Mc85] E.M. McCreight: "Priority Search Tree," SIAM J. Comput., 14, 2, pp.257-276, 1985.

[OS84] T. Ohtsuki and M. Sato: "Gridless Routers for Two-Layers Interconnection," Proc. of ICCAD84, Santa Clara, pp.76-79, 1984.

[ST084] M. Sato, M. Tachibana, and T. Ohtsuki: "An Algorithm for Resizing Polygonal Regions and Its Applications to LSI Mask Pattern Design," Electronics and Communications in Japan, 67-C, 4, pp.93-101, 1984. Translated from Trans. IECE Japan , 66-c, 12, pp.1132-1139, 1983.

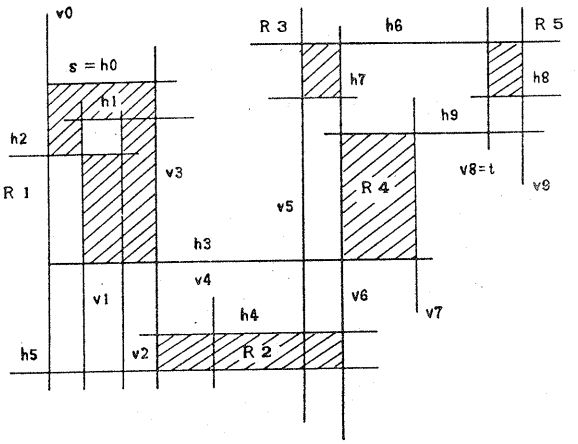

Fig. 1. Generalized Manhattan path problem defined by 10 horizontal and 10 vertical line segments and 5 connected regions.

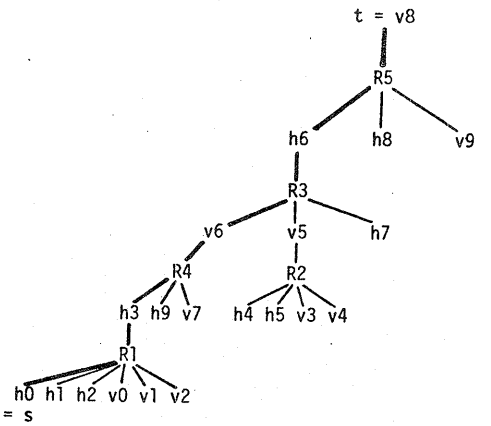

Fig. 2. Search tree constructed by the algorithm.

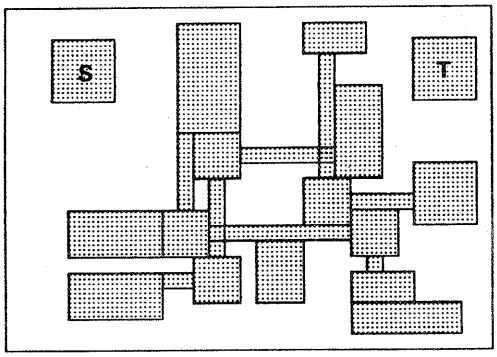

Fig. 3. Initial layout pattern. S and T are terminals to be connected.

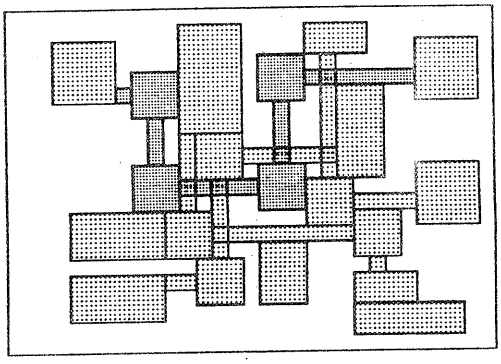

Fig. 4. Resulting layout pattern.

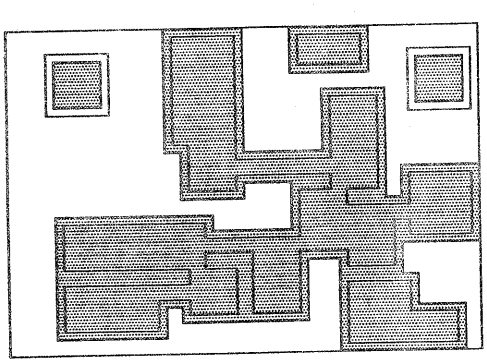

Fig. 5. Horizontal Routable region HR(S,T,X,w_y).

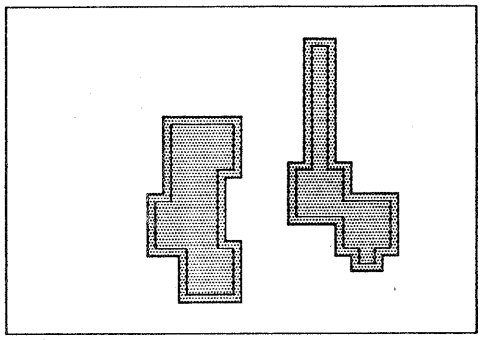

Fig. 6. Vertical Routable region VR(S,T,Y,w_x).

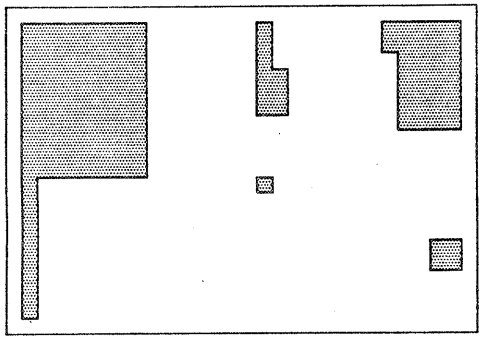

Fig. 7. Via acceptable region VA(S,T,X,Y,d_x,d_y).

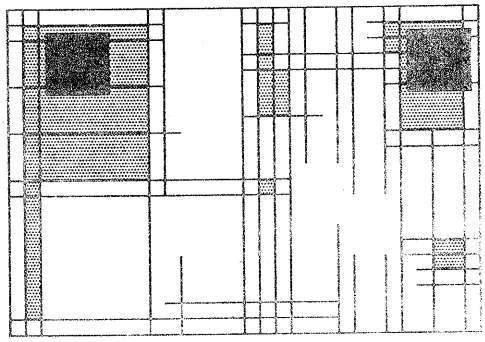

Fig. 8. Sets of line segments and Via Acceptable region.