Euclidean Shortest Path Problem with Rectilinear Obstacles[1]

Joon Shik Lim, S. Sitharama Iyengar, and Si-Qing Zheng

Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803-4020

Abstract

This paper presents new heuristic algorithms using the guided A^* search method : Guided Minimum Detour (GMD) algorithm and Line-by-Line Guided Minimum Detour (LGMD) algorithm for finding optimal rectilinear (L_1) shortest paths in the presence of rectilinear obstacles. The GMD algorithm runs O(nN(logN) + tN) in time and takes O(N) in space, where N is the number of extended line segments including boundary of the obstacle; and t is the number of intersected boundary of the obstacles on the trial or the escape line in n ✗n grid graphs. In the LGMD algorithm, we derive an O(N(logN) + tN) time and O(N) space algorithm in gridless graphs.

1. Introduction and overview

There are two basic classes of sequential algorithms: *maze-running* algorithms and *line-search* algorithms. These algorithms are mostly aimed at finding an obstacle-avoiding path, preferably the shortest one, between two given points on the routing space. The first such algorithm is the *Lee* algorithm [11], which is an application of the *breadth-first* shortest path search algorithm. There are a large number of variations [1, 6,7, 10, 15-18, 20, 21] of the original Lee algorithm.

The line search algorithms have been proposed to improve the performance. The first of such algorithms is reported in [8] and [13]. Several recent line search algorithms [4, 14, 19, 22] are based on powerful computational geometry techniques. Wu et al. [22] introduce a rather small connection graph, *track graph*, that contains the shortest path but it is not strong. For the n points shortest paths problem, the run time is $O((e + k)\log t)$, where e is the total number of edges, t is the extreme edges of all obstacles, and k is the number of intersections among obstacle tracks, which is bounded by $O(t^2)$. De Rezende et al. [19] construct a strong connection graph in time $O(n\log n)$ with rectangle obstacles. Clarkson et al. [4] generalize the shortest problem to the case of arbitrarily-shaped obstacles in time $O(n(\log n)^2)$. When obstacles are just rectilinear line segments, Berg et al. [2] studies the shortest path problem in a combined metric that generalizes the L_1 metric and the rectilinear link metric.

In this paper, we introduce new algorithms, the *Guided Minimum Detour* (*GMD*) algorithm and the *Line-by-Line Guided Minimum Detour* (*LGMD*) algorithm. Our *GMD* algorithm uses a heuristic, *guided* A^*, search method. Also we present another modified algorithm, the *LGMD* algorithm, using a *priority queue* to select the

[1]This project is in part funded by LEQFS-RD-A04 grant. Feb. 1, 1992.

line segment that has the lowest detour length and some superimposed data structure.

2. The Guided Minimum Detour algorithm

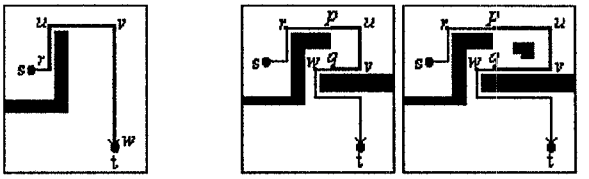
Let E be a *Jordan curve* consisting of n axis parallel line segments that contain no two consecutive line segments which are collinear. Let B be a set of obstacles each of which is a simple rectilinear polygon consisting of E and its interior without holes in the interior. We use G_p to denote the partial grid obtained by deleting all grid nodes covered by obstacles of B. We assume that a path P is a set of E that consists of a finite number of axis parallel line segments in G_p such that a directed path P is $(v_1, v_2, \ldots, v_m)$ from v_1 to v_m in G_p i.e., $P = (v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_m)$ with vertex set $\{v_1, v_2, \ldots, v_m\}$ and edge set $\{(v_i, v_{i+1}): i = 1, \ldots, m\text{-}1\}$. If a grid node v_1 neighbors to v_2, the line segment $v_1 \rightarrow v_2$ is called a *unit line segment*. The length of P, denoted by $\lambda(P)$, is the number of the unit line segments in P. A grid node c of G_p is called a *critical node* if one of the following conditions holds: (i) c is the source node s, (ii) c is a boundary node of G_p, and (iii) c is on a horizontal or vertical line that passes through the target node t.

For the shortest path from s to t in G_p, the *Manhattan distance* $M(s, t)$ between s and t is used as a lower bound. The *detour length* of a path P, denoted by $DL(P)$, from s to t is the total number of grid nodes that proceed away from the target node t in P. The following theorem shows the properties of the shortest path in *Minimum Detour* (*MD*) algorithm in [7].

Theorem 1 [7].

1. *A path* $P = (s \rightarrow \ldots \rightarrow t)$ *has length* $\lambda(P) = M(s, t) + 2 \cdot DL(s, t)$.
2. *Let* P' *be a subpath of* P *from* s *to* x *with* $DL(s, x)$. *Then* $\lambda(P') = M(s, t) + 2 \cdot DL(s, x) - M(x, t)$.
3. *If* $DL(P)$ *is minimized, then* $\lambda(P)$ *is the shortest path from* s *to* t.
4. *The path generated by MD is the shortest one with the minimized* $DL(s, t)$.

A subpath $D = (r \rightarrow u \rightarrow v \rightarrow w)$ in P is called a *detour* (Fig. 1.a), if the directions of the consecutive line segments $r \rightarrow u$, $u \rightarrow v$, and $v \rightarrow w$ are different, where $u \rightarrow v$ consists of one or more than one line segment with the same direction. We say that a detour D is reducible if (i) there exists a detour $R = (p \rightarrow u \rightarrow v \rightarrow q)$ of D where $r \leq p < u$, $v < q \leq w$, and $\lambda(p \rightarrow u) = \lambda(v \rightarrow q)$; and (ii) the rectangle composed by corner nodes (p, u, v, and q) is the largest non-zero area and the edge (p, q) does not intersect any obstacle. The path R is called *reducible detour* (Fig. 1.b).



a. A Detour ($r \rightarrow u \rightarrow v \rightarrow w$) b. Reducible Detours ($p \rightarrow u \rightarrow v \rightarrow q$)

Fig. 1 Detours

Each line segment l is extended without changing its direction until it hits a critical node c from a non-critical node, or reaches a corner critical node c from a critical node on a same boundary line segment. We call such a critical node c as a *base node*. We also classify the source node s as a base node.

We maintain three priority queues: *NEW*, *OLD*, and *TEMP* of line segments; the line segments to be extended for next iteration are stored in *NEW*; the line segments that have already been extended from the previous iteration are stored in *OLD*; and the currently extending line segments are stored in *TEMP*. Also, using some superimposed data structures to be described in section 2 we maintain two sets of line segments: (i) *COMPLETE* contains line segments that have already been extended; (ii) *CRITICAL* is the set of critical nodes represented by line segments. Fig. 2 shows examples solved by the *GMD* algorithm.

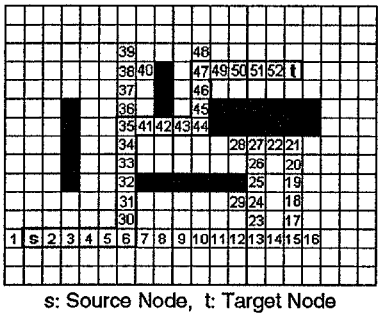


Fig. 2 Examples for The *GMD* Algorithm

By the Theorem 2, we show that the size of the searched space generated by the *GMD* algorithm is less than the one of the searched space by *MD* algorithm, which is the primary advantage of our *GMD* algorithm. In order to assure that the path found is the shortest one, we need to show that a path P generated by the *GMD* algorithm has the minimized detour length $DL(P)$ by the Theorem 3.

Theorem 2. *The set of searched nodes by the GMD algorithm is a subset of the one by the MD algorithm.*

Theorem 3. *The path P=(s*fi . . . fi *t) generated by the GMD algorithm is the shortest path.*

The data structures realizing the *GMD* algorithm are the next element subdivision of *CRITICAL* (the set of N_c line segments) with some superimposed structure [5] and the set of extended line segments *COMPLETE* (the set of N_e line segments) with the *priority search tree* [12]. The former can be built in $O(N_c \log N_c)$ and the latter can be done in $O(N_e \log N_e)$.

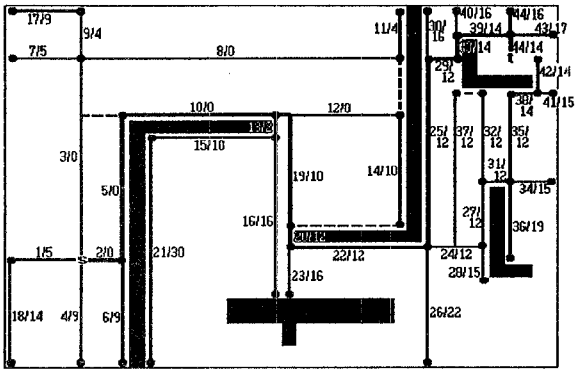
The line segments in *COMPLETE* are grouped to co-horizontal and co-vertical line segments and each group is stored into the priority search tree. Whenever a line segment is extended to a neighbor node, the node should be investigated whether it is visited, base node, or not. This operation can be done in $O(\log N_e + \log N_c)$ time using $O(N_e + N_c)$ space by the Theorems 4 and 5. If N is $N_e + N_c$, the operation can be done in $O(\log N)$ time using $O(N)$ space.

Theorem 4 [5]. *The t elements that intersect a query line segment can be found in O(logN + t) time with a data structure that uses O(N) space. This structure can be constructed in O(NlogN) time.*

Theorem 5 [12]. *For a set of line segments and a query line segment, the first point that intersects the query line segment can be found in O(logN) time with the priority search tree that uses O(N) space. This structure can be constructed in O(NlogN) time. A new line segment can be added or an old one deleted in O(NlogN) time.*

3. A modified algorithm: line-by-line Guided Minimum Detour algorithm

Let us now consider another modification of our *GMD* algorithm. Without loss of general features of the *GMD* algorithm, we contemplate *line-by-line* extensions to generate a line segment instead of node-by-node extensions. Each line segment in *COMPLETE* must be from a base node to a base node except the line segment constructed by deleting reducible detour. In other words, all alive line segments in *OLD* are extended until base nodes are hit regardless their lower bound. The line segment that has the lowest detour length will be chosen for the next extensions. To implement this modification, we use a *priority queue*, called *OPEN*, to select the line segment that has the lowest detour length. By the queue *OPEN*, the global variable d, detour length, in the *GMD* algorithm is not needed. Such a modified algorithm is called *Line-by-Line Guided Minimum Detour* (*LGMD*) algorithm. Unlike the *GMD* algorithm, the *LGMD* algorithm does not need grid graph but directly consider polygonal regions as space for routing. Moreover, the algorithm not only compromises the existing the *GMD* algorithm's drawback--the running time--but also shares the optimality of the *GMD* algorithm. The generated whole line segments with sequence numbers and detour lengths are shown in Fig. 3. A comparison of new algorithm with the existing algorithms is presented in the following Fig. 4.



- - - :Trial Line for Deleting Reducible Detour, • : Base Node, A/B : A - Order of Extensions, B - Detour Length

Fig. 3 Extended Line Segments for The *LGMD*

	Lee	*Hadlock*	*Wu et al.*	*Lim, Iyengar, and Zheng (GMD)*	*Lim, Iyengar, and Zheng (LGMD)*
Time	$O(n^2)$	$O(n^2)$	$O(n\log n)$	$O(nN(\log N) + tN)$	$O(N(\log N) + tN)$
Space	$O(n^2)$	$O(n^2)$	$O(n)$	$O(N)$	$O(N)$
Search Method	Breadth -First	A^*	Dijkstra's Search on Track Graph	Grid-by-Grid Guided A^*	Line-by-Line Guided A^*
Optimality	Optimal	Optimal	Suboptimal	Optimal	Optimal

Fig. 4 Bounds on the Algorithms

4. Conclusions

We introduced a heuristic approach to find an optimal rectilinear (L_1) shortest path. Our *GMD* and *LGMD* algorithms always find an optimal rectilinear shortest path using the *guided* A^* search method without constructing a connection graph. The *GMD* algorithm runs $O(nN(\log N) + tN)$ in time and takes $O(N)$ in space in n x n grid graphs. Also, we derive an O(N(logN) + tN) time and $O(N)$ space algorithm, the *LGMD* algorithm, in n x n gridless graphs.

Since the detour length as a lower bound in the *GMD* or the *LGMD* algorithm can be substituted for the number of bends in the rectilinear link (L_1) metric [2] or the channel wiring density [3], our algorithms can be easily extended to the problems both L_1 metric and the channel wiring density without losing the optimality. Moreover, if we combine the lower bounds--detour length, number of bends, and/or channel wiring density--by some control strategies, the algorithms could be applied to smooth out the wiring density with the minimum-bend shortest path.

5. References

[1] S. B. Akers, "*A Modification of Lee's Path Connection Algorithm*," IEEE Transactions on Elec. Computers, EC-16(2), pp. 97-98, 1967.

[2] M. T. De Berg, M. J. Van Kreveld, B. J. Nilsson, and M. H. Overmars, "*Finding Shortest Paths in the Presence of Orthogonal Obstacles Using a Combined L_1 and Link Metric*," In Proceedings of the 2nd Scandinavian Workshop on Alg. Theory, pp. 213-24, 1990.

[3] A. D. Brown and M. Zwolinski, "*Lee Ruter Modified for Global Routing*," CAD, Vol. 22, No. 5, pp. 296-300, June, 1990.

[4] K. L. Clarkson, S. Kapoor, and P. M. Vaidya, "*Rectilinear Shortest Paths through Polygonal Obstacles in $O(n(\log n)^2)$ Time*," in Proceedings of the Third Annual Conference on Computational Geometry, pp. 251-57, ACM, 1987.

[5] H. Edelsbrunner, M. H. Overmars, and R. Seidel, "*Some Methods of Computational Geometry Applied to Computer Graphics*," Compter Vision, Graphics, and Image Processing, Vol. 28, pp. 92-108, 1984.

[6] J. M. Geyer, "*Connection Routing Algorithms for Printed Circuit Boards*," IEEE Tran. on Circuit Theory, CT-18(1), pp. 95-100, 1971.

[7] F. O. Hadlock, "*The shortest Path Algorithm for Grid Graphs*," Networks, 7, pp. 323-34, 1977.

[8] D. W. Hightower, "*A Solution to Line Routing Problems on the Continuous Plane*," in Proceedings of the Sixth Design Automation Workshop, pp. 1-24, IEEE, 1969.

[9] P. Hart, N. Nilsson and B. Raphael, "*A Formal Basis for the Heuristic Determination of Minimum Cost Paths*," IEEE Transactions on Systems, Science and Cybernetics, SCC-4(2), pp. 100-107, 1968.

[10] J. H. Hoel, "*Some Variation of Lee's Algorithm*," IEEE Transactions on Computers, C-25(1), pp. 19-24, 1976.

[11] C. Y. Lee, "*An Algorithm for Path Connections and Its Applications*," IRE Tran. on Elec. Comp., EC-10(3), pp. 346-65, 1961.

[12] E. M. McCreight, "*Priority Search Trees*," SIAM J. Comput., Vol. 14, No. 2, pp. 257-76, May 1985.

[13] K. Mikami, K. Tabuchi, "*A Computer Program for Optimal Routing of Printed Circuit Connectors*," IFIPS Proc., H-47, pp. 1475-78, 1968.

[14] J. S. B. Mitchell, "*An Optimal Algorithm for Shortest Rectilinear Paths Among Obstacles in the Plane*," In Abstacts of the First Canadian Conference on Computational Geometry, p. 22, 1989.

[15] E. F. Moore, "The Shortest Path Through a Maze," Annals of the Harvard Computation Laboratory, Vol. 30, Pt. II, pp. 285-92, 1959.

[16] T. Ohtsuki, "*maze-running and Line-search Algorithms*," in T. Ohtsuki, editor, Advances in CAD for VLSI, Vol. 4: Layout Design and Verification, pp. 99-131, North-Holland, New York, 1986.

[17] I. Pohl, "*Heuristic Search viewed as path finding in a Graph*," Artificial Intelligence, Vol. 1, pp. 193-204, 1970.

[18] ___, "*Bi-Directional Search*," Machine Intelligence, Vol. 6, pp. 127-40, 1971.

[19] P. J. Rezend, D. T. Lee, and Y.-F. Wu, "Rectilinear Shortest Paths with Rectangular Barriers," in Proceedings of the Second Annual Conference on Computational Geometry, pp. 204-13, ACM, 1985.

[20] F. Rubin, "*The Lee Path Connection Algorithm*," IEEE Transactions on Computers, C-23(9), pp. 907-14, 1974.

[21] J. Soukup, "*Fast Maze Router*," in Proceedings 15th Design Automation Conference, pp. 100-102, 1978.

[22] Y.-F. Wu, P. Widmayer, M. D. F. Schlag, C. K. Wong, "*Rectilinear Shortest Paths and Minimum Spanning Trees in the Presence of Rectilinear Obstacles*," IEEE Transactions on Computers, C-36(3), pp. 321-31, 1987.