Improved Lee's Algorithm on Electronic Maps

Gene Eu Jan[1] and Ki-Yin Chang[2]

[1]*Department of Computer Science*

[2]*Department of Merchant Marine*

National Taiwan Ocean University,

Keelung, Taiwan

Abstract

This paper presents a routing algorithm with linear time complexity for computing the shortest path between a given set of cells on a digital electronic map (raster plane). By incorporating the *nuclear fission chain reaction* scheme, the path connection confined to a rectilinear movement in Lee's algorithm, probably the most widely used method for finding paths on printed circuited boards and planning robot paths on grid spaces, is therefore overcome by the proposed method. Instead of Lee's rectilinear or staircase approach, an oblique line method is employed to obtain the shortest path of two planar cells that is much closer to the actual path on the geographical map. This method can be extended to find the shortest paths for q vessels with the time complexity of $O(qN)$, where N is the number of cells in the raster electronic charts.

Key Words: electronic maps, path routing, nuclear fission chain reaction.

1. INTRODUCTION

In recent years, digital electronic maps (DEM) and ECDIS (electronic chart display information systems) have been widely used in marine navigation, GPS applications and geographic information systems. Finding the shortest path from

any origin, without crossing any barriers, to all destinations in a digital electronic map is of great significance. In ECDIS marine navigation, the most common application is to determine the shortest or the most economical path leading from any original point (cell, port) to the final destination without crossing any landmass (also called obstacle or barrier) including shoals.

Quite a few researchers have explored this field. Lee presented a shortest-route algorithm using planar cells applied to electronic circuit arrangements, maze games and raster map search problems [1][2][3]. Lee's algorithm introduced a series of 4 cell-connected neighborhoods to determine the shortest path between the start and end points. Two linked lists L and L_1 are defined to speed up the search and trace procedures. In this method the cell list L is an ordered list containing the cell names. The auxiliary list L_1 is provided for momentary cell name storage. Lee's algorithm has space and time complexities of $O(N)$ for the determining the shortest path between two cells. The deficiency in Lee's algorithm is that it is implemented onto a set of square cells in which equal distances are constrained between each cell and its' neighborhood. The cells in his algorithm are therefore limited to extension in only four directions and the shortest path is thus limited to a rectilinear instead of an oblique line.

The endeavor in the staircase approach in the free space has never been suspended since Lee's presentation. Recently, Jan [4] applied a *radiation* scheme to the raster map. This method is more extensive in searching for the shortest route because an 8 cell-connected neighborhood is used with diagonal cells. However, this algorithm increases the time complexity to $O(N^2)$ with the same space complexity.

The shortest path algorithm presented in this paper is a substantial

improvement of Jan's previous algorithm by reducing the time complexity to $O(N)$ with a combination of the *nuclear fission chain reaction* method and an extra data structure. Our algorithm was developed from an idea that stemmed from the process involving neutrons hitting neighboring atoms (neighboring cells) with each neighboring atom releasing new neutrons. This chain reaction spreads into the entire space until every cell has been split exactly one time. This process is terminated when all of the atoms have released their neutrons.

The earliest work involving this problem was the displacement of an autonomous vehicle on Mars produced by Kirk and Lim [5]. In their attempts to solve this problem, Larson and Sadiq [6] considered the case of regions forbidden only to travel in the case of Manhattan distance. Viegas and Hansen [7] studied the problem of determining the shortest path between a given set of origin (or source) and destination points with polygonal barriers and concluded that the obstacles could be open polygonal line segments and /or not necessarily convex polygons in the two-dimensional plane. Chen and Ramanan [8] improved the Euclidean shortest path in the presence of obstacles. Recently, Fagerholt *et al* [9] presented a vector model for solving a shortest path with obstacles based on Dijkstra's algorithm [10]. Their worst-case time complexity was $O(N^2)$. However, all of the vector-based algorithms are still poor in solving cases involving complicated concave obstacles.

The Lee's algorithm is considered as a wave propagation problem. It has been widely adopted for obstacle avoidance of robot path planning on grid spaces, but it is confined to a rectilinear movement. Several approaches have been proposed to solve robot path planning problems [11-14]. Amount them, Diaz *et al* [14] have recently proposed a method using both the *fast distance transformation* (FDT) and some topological methods to obtain free space skeleton. The improved Lee's

algorithm presented in this paper can be however applied to the diagonal movement to obtain the shortest path of two planar cells with the time complexity of $O(N)$. This method is capable of finding the shortest path from any source cell to any destination cell with all sorts of complicated obstacles or even in maze.

2. RASTER DATA STRUCTURES

An electronic chart data structure can be divided into two classes: vector structure and raster structure or *cellular organization* structure. The raster technique is more attractive for two reasons. First, complete areas can be filled in easily with a raster technique by finding and turning "on" the pixels that are inside the boundary of an area. In vector graphics, a "crosshatch pen" must be used to produce shading. Second, the full spectrum of color is easily obtained, while a vector display is limited to the number of pens available for a pen plotter [15].

Compared with vector graphics, raster graphics has a simpler data structure that requires less computation [16][17]. We therefore propose a new algorithm for the shortest path problem on raster electronic charts.

Although the raster structure is particularly suitable for earth surface applications, it has some limitations. The cells adjacent to any cell may not be evenly spaced, depending on their relative positions to the center cell. For example, the cells above, below and on the left/right of the center cell are 1 unit of distance from the center cell. The distance of the cells on the diagonal are $\sqrt{2}$ units from the center, as shown in Figure 1. Figure 2 depicts two cell connection styles. The number of decimal digits in the irrational number, $\sqrt{2}$, is determined by the total number of raster cells. For a 4 cell-connected neighborhood, depicted in Figure 2(a), we are interested in the cells above, below, and on the left/right of a cell only when

all of the cells are equidistant from the center cell. The neighboring cells thus share an edge. If the four cells on the diagonal are included, we are working in an 8 cell-connected neighborhood, as depicted in Figure 2(b). The cells in this arrangement are not evenly spaced such that the diagonal cells in the neighborhood share only a vertex, while the others share an edge. Because all of the cells in these two connection styles have neighbors of the same size and shape, we say that they have *spatial neighborhood similarity* [18].

$2\sqrt{2}$	$1+\sqrt{2}$	2	$1+\sqrt{2}$	$2\sqrt{2}$
$1+\sqrt{2}$	$\sqrt{2}$	1	$\sqrt{2}$	$1+\sqrt{2}$
2	1	0	1	2
$1+\sqrt{2}$	$\sqrt{2}$	1	$\sqrt{2}$	$1+\sqrt{2}$
$2\sqrt{2}$	$1+\sqrt{2}$	2	$1+\sqrt{2}$	$2\sqrt{2}$

Figure 1. Illustration of the distances from the center cell to neighboring cells on a raster electronic chart.

	Neighbor	
Neighbor	Center Cell	Neighbor
	Neighbor	

(a) A 4 cell-connected neighborhood.

Diagonal Neighbor	Neighbor	Diagonal Neighbor
Neighbor	Center Cell	Neighbor
Diagonal Neighbor	Neighbor	Diagonal Neighbor

(b) An 8 cell-connected neighborhood.

Figure 2. Cells connection styles.

A given black-and-white geometric map, where the black and white areas represent the land and the sea, respectively, can be converted into a rasterized graphic by a scanning process. The final data is then stored in a computer. There should not be any significant distortion between the original geometric map and the final computerized raster because the size and shape of the cells are exactly the same as the pixels.

3. THE SHORTEST PATH ALGORITHMS

A navigational track is affected by many natural factors, such as current, wind, weather, water density, etc. This study developed an algorithm for the simplest case, that is, to determine the shortest route between any two points (assumed to be ports) without any natural factor influences. Natural factor influences will be considered in our future studies.

3.1 THE DATA STRUCTURE OF MAP

In formatting of the cell structure of an electronic chart, the number of data fields is flexible for meeting the requirements of the electronic map. There are three parameters for cell storage, that is, *S/L*, *AT* and *Vis*. The *S/L* (*Sea or Land*) parameter distinguishes whether a cell is a landmass (the value is infinity) or navigable area (the value is one). The *AT* (time of arrival) parameter stores the time needed to travel from the source cell to the current cell and its initial value is infinity. The third parameter *Vis* (*Visited*) distinguishes whether the cell has been visited or not and its initial Boolean value is *false*. The initial cell conditions are illustrated in Figure 3, where the black cells represent landmasses and the white cells represent navigable areas.

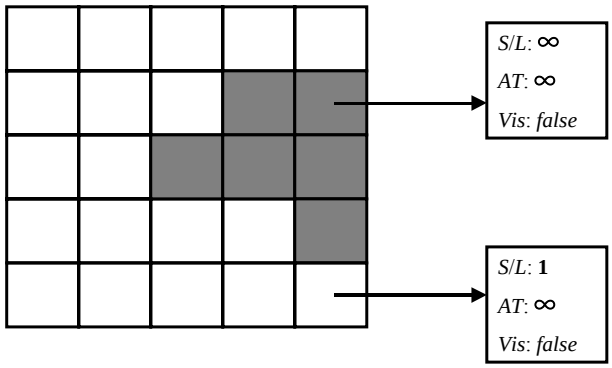


Figure 3. Cells data structure.

In an $m \times n$ raster map, any cell $C_{i,j}$ has three parameters $S/L_{i,j}$, $AT_{i,j}$ and $Vis_{i,j}$, where $1 \leq i \leq m$, $1 \leq j \leq n$.

3.2 THE STRUCTURE OF A PARTICULAR VOYAGE

The data structure of a particular voyage for this algorithm, as shown in Figure 4, includes the particular voyage (or vessel path), the source cell, the destination cell, the speed, etc. The speed field stores the vessel speed, which is the number of cells traveled by the vessel per unit time. The vessel path is stored in a linked list in which each node represents a cell in the path. Each node of the linked list for the path contains four fields, *Row*, *Col*, *AT* and *Next*. The *Row* and *Col* field store each cell's coordinates, *i* and *j*, respectively. The *AT* field stores the $AT_{i,j}$ value of the cell $C_{i,j}$. The *Next* field is a link that points to the next node. By modifying the value of the *AT* field in the linked list of the path, we can adjust the vessel speed .To illustrate multiple vessels with different speeds on one chart, the *AT* field values should be modified. The *AT* field detail for the plural pairs shortest path-searching algorithm will be discussed in Section 3.5. For a vessel in a one pair shortest path, variations in speed are ignored to simplify the case.

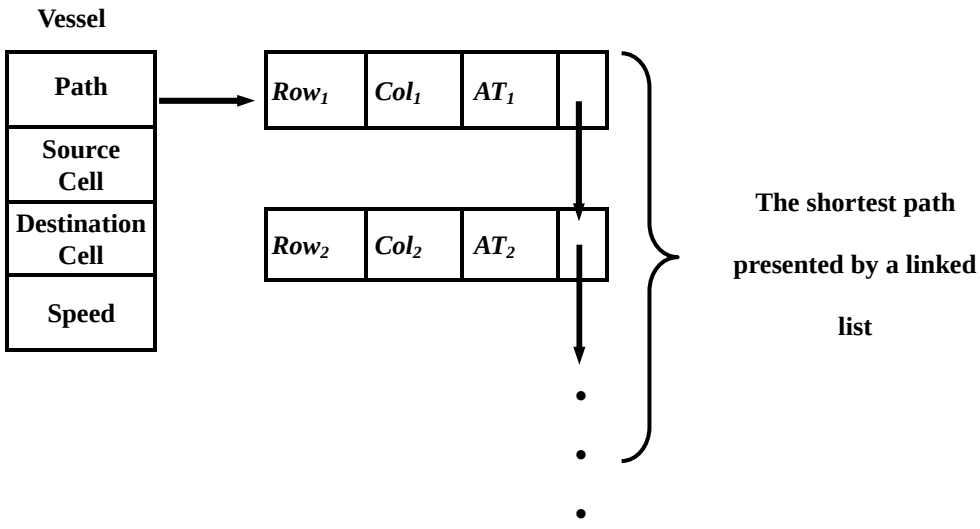


Figure 4. The data structure of a particular voyage.

If a vessel travels from the source cell, located at (2, 2), to the destination cell (4, 3) through the cell (3, 2), the data fields (*Row*, *Col*, *AT*) in the first node of its linked list can be represented as (3, 2, 1), as shown in Figure 5.

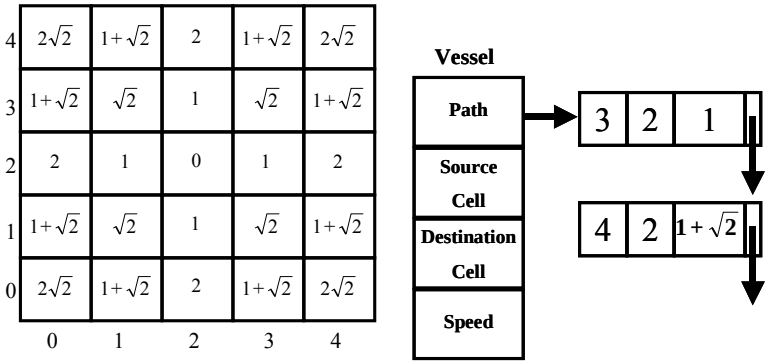
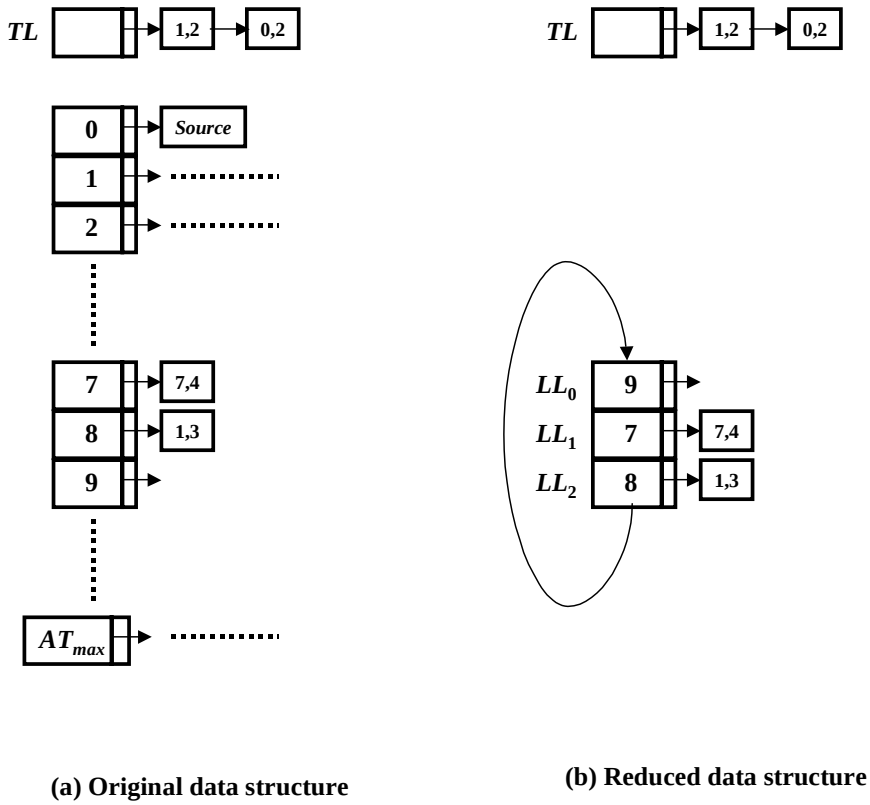


Figure 5. An illustration of the data structure of a particular voyage.

3.3 THE DATA STRUCTURE OF ALGORITHM

The data structure of this algorithm is the buckets ß that consists of a sequence of initially empty queues with continuous integers in the range 0 to AT_{max}, where AT_{max} is the maximum *AT* and obviously less than *N*. Each queue is called a bucket LL_l that stores a series of cell indices, where *l* is both the index and the header value of bucket LL_l and $l \le AT_{i,j} < l+1$ for the $AT_{i,j,}$ values of the cells in the bucket LL_l.

The actual *AT* values for cells in the map data structure (considered as a matrix) can be easily accessed using a sequential allocation equation. For an $m \times n$ raster chart, the first step in the algorithm is to identify a source cell (expressed by *S*) and a destination cell (expressed by *D*). According to the algorithm, the *AT* value of the source cell is 0 and the location of the source cell *S* will be moved into the first bucket, LL_0. The remaining cells that spread from *S* will be moved into their corresponding buckets according to their *AT* values. The *AT* value of any cell would be increased at most by $\sqrt{2}$ from its neighboring cell. Therefore, the number of buckets can be reduced to 3 (LL_0, LL_1, LL_2) for the purpose of recycling since the updated cells would be moved into one of the next two buckets. A temporary bucket *TL* is applied to the algorithm to store the indices (*i*, *j*) of visited cells until its final value is obtained once all of the cells in the current bucket have been processed. An example of the buckets for the original and reduced data structure is depicted in Figure 6(a) and 6(b), respectively.



(a) Original data structure **(b) Reduced data structure**

Figure 6. The data structures of algorithm.

3.3 THE SHORTEST PATH ALGORITHM FOR A SINGLE PAIR

In order to reduce the memory space, the ß buckets are replaced with three circular buckets. The three buckets marked as LL_{index}, where the index is an integer variable and $0 \leq index \leq 2$.

We also defined a function named Update_*AT&Vis* to support the structured program. The purpose of the function Update_*AT&Vis* is to update the $AT_{i,j}$ and $Vis_{i,j}$ values of $C_{i,j}$. There are two input parameters in this function named (i, j) and $new\_AT_{i,j}$, the updated $AT_{i,j}$.

Function Update_*AT&Vis* $((i, j), new\_AT_{i,j})$

Step 1: Decide if the indices of $C_{i,j}$ (also called (i, j)) should be moved into a temporary bucket TL .

If $Vis_{i,j}$ is *false,* then the Boolean value is updated to *true* and the $C_{i,j}$ indices are moved into the temporary bucket TL.

Step 2: Access to $C_{i,j}$ in the memory is obtained using a sequential allocation equation $Loc(i, j) = Loc(0, 0) + n*i + j$, to update its $AT_{i,j}$ value for a $m \times n$ raster map, where $Loc(i, j)$ is the memory location of $C_{i,j}$ in a 2D array.

If $new\_AT_{i,j}$ is less than $AT_{i,j}$, then $AT_{i,j} = new\_AT_{i,j}$.

END {Function of Update_*AT&Vis* }

Algorithm 1: The shortest path

Initialization:

For each cell, $C_{i,j}$ $(S/L_{i,j}, AT_{i,j}, Vis_{i,j})$ in an $m \times n$ raster chart, the initial $S/L_{i,j}$ value is one if cell $C_{i,j}$ is in the sea area or infinity if it is in the landmass. $AT_{i,j} = \infty$ and $Vis_{i,j}$ = false for all cells, where $1 \leq i \leq m$, $1 \leq j \leq n$.

The initial value of the index is 0.

Step 1:

Step1.1: Identify a source cell *S*. If the source cell $S/L_{i,j}$ = 1 then update $AT_{i,j}$ = 0 otherwise return the error message "The source that you spotted is in the landmass".

Step 1.2: Identify a destination cell *D*. If the destination cell's $S/L_{i,j} = \infty$ then return the error message "The destination that you spotted is in the landmass".

Step 1.3: Set the speed unit of the vessel to the cell/unit time.

Step 2: Compute the time of arrival between the source cell and the remaining cells.

Step 2.1: Move the source cell *S* into the temporary bucket *TL* and update the source cell's $Vis_{i,j}$ to *true.*

Step 2.2: Move the source cell *S* into the bucket LL_0.

Step 2.3: For each cell in the LL_{index}, update the $AT_{i,j}$ of its neighboring cells.

Step 2.3.1: Remove the indices of the first cell $C_{i,j}$ from the front end of the LL_{index}.

Step 2.3.2: Update the time of arrival of the 8 cell-connected neighbors of cell $C_{i,j}$.

For each $C_{i,j}$'s neighbor $C_{i',j'}$, if the cell's *S/L*=1 then call Update_*AT&Vis* $((i', j'), AT_{i,j}+1)$ for the 4 cell-connected neighbors; otherwise call Update_*AT&Vis* $((i', j'),$

$AT_{i,j} + \sqrt{2}$) for the four diagonal neighbors.

Step 2.3.3: Iterations

If LL_{index} is not empty, then repeat steps 2.3.

Step 2.4: Move the cells' indices in the *TL* into their corresponding buckets.

Step 2.4.1: For all the indices in the *TL*, move (*i*, *j*) from *TL* into $LL_{\lfloor AT_{i,j} \rfloor \bmod 3}$

Step 2.4.2: If the *TL* is empty, then update the *index* value

index = (*index*+1) mod 3.

Step 2.5: Iterations

If two consecutive buckets are not empty, then repeat step 2.3.

Step 3: Backtracking

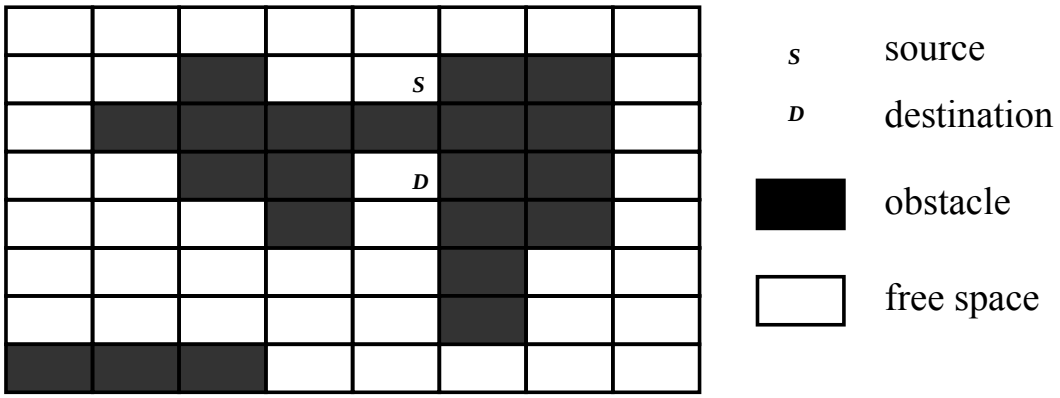
If the $AT_{i,j}$ value of the destination cell is infinity, return the error message "There is no path between the source cell and the destination cell"; otherwise backtrack the shortest path from the destination cell by selecting one of the 8 cell-connected neighboring cells with the smallest time of arrival value and repeating the selection step by step until the source cell is reached.

Step 4: Reversing the path

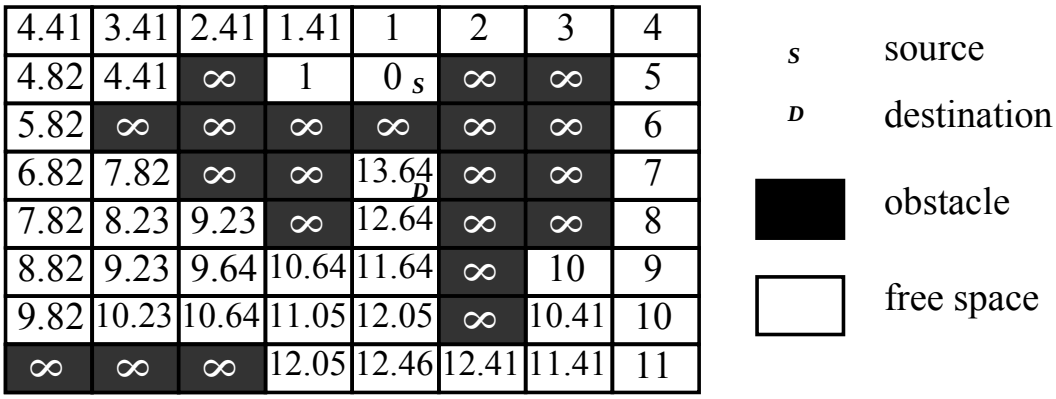
Reverse the path derived from step 3 to obtain the desired shortest path.

END {Algorithm of the shortest path}

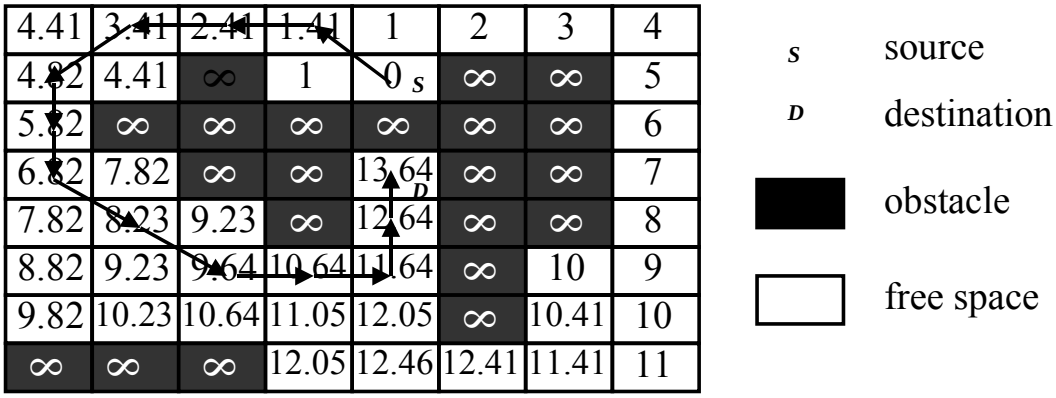
To explain the detailed computation of the shortest path algorithm for a single pair, an example is illustrated in figure 7.



(a) Initialization.



(b) Execution result of step 1 and step 2.



(c) Backtrack the shortest path from destination to source, then reverse the path derived from the backtrack step to obtain the desired shortest path.

Figure 7. Illustration of the shortest path algorithm on an 8 × 8 raster.

If the improved Lee's path routing is limited to only four rectilinear directions, the routing problem is therefore characterized as a Lee's path algorithm. Compare the case of the Lee's algorithm shown in Figure 8 with the improved Lee's algorithm shown in Figure 7(c), the latter algorithm obviously shortens the distance of the connection from source to destination.

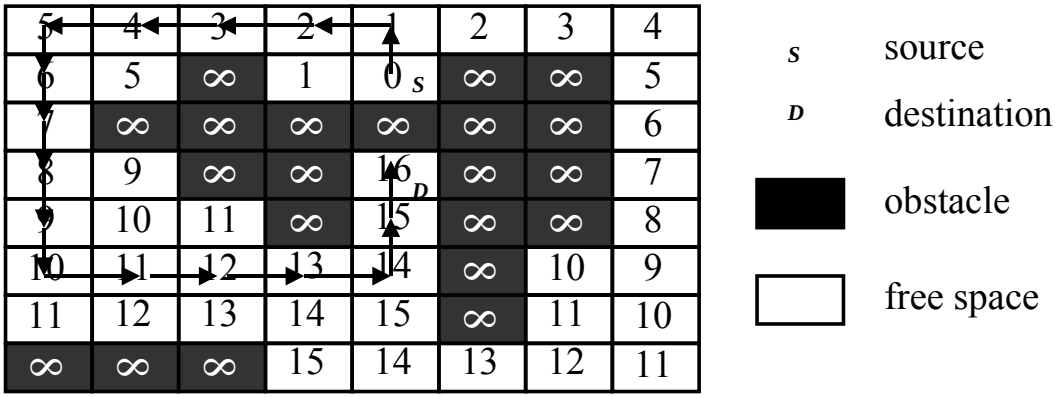


Figure 8. Illustration of the Lee's algorithm on an 8 × 8 raster.

3.4 PERFORMANCE ANALYSIS

The following lemma is applied to prove the correctness of this algorithm, that is, the path obtained from this algorithm is indeed the shortest.

Observation 1: Once all cells of the LL_{index} have updated the $AT_{i,j}$ of its neighboring cells and the LL_{index} is empty, all cells in the TL should be moved into either $LL_{(index+1) \bmod 3}$ or $LL_{(index+2) \bmod 3}$ according to their $AT_{i,j}$ values.

According to steps 2.3.2 and 2.3.3, the minimum $AT_{i,j}$ value of cells in the TL would be no less than (the minimum $AT_{i,j}$ value in the LL_{index})+1 and the maximum $AT_{i,j}$ value of cells in the TL would be no greater than (the maximum $AT_{i,j}$ value in the LL_{index})+$\sqrt{2}$. The LL_{index} is empty when step 2.3 is completed. After the computation of steps

2.4.1, all of the cells in the temporary bucket TL should be moved into either $LL_{(index+1) \bmod 3}$ or $LL_{(index+2) \bmod 3}$ according to their $AT_{i,j}$ values.

Observation 2: All of the previous buckets should be empty in step 2.3.

Lemma 1: The AT_j values of cells in the LL_{index} will be minimized in step 2.3.

Proof: This lemma is proven by its contradiction. For the $AT_{i,j}$ value of any cell $C_{i,j}$ in the current bucket LL_c, where $0 \le c \le AT_{max}$, we assume that there exists a smaller $AT_{i,j}'$ value updated by another cell $C_{i',j'}$ that has $AT_{i',j'}$ value. According to step 2.3.2, the minimum value of cell $C_{i,j}$, $AT_{i,j}$, should be replaced by $AT_{i,j}'$ if the cell $C_{i',j'}$ is in one of the previous buckets LL_l where $0 \le l \le c$-1. If the cell $C_{i',j'}$ is in the current or one of the higher buckets LL_l, where $c \le l \le AT_{max}$, we know that $c \le AT_{i,j} < c+1$, $c \le AT_{i',j'}$. $AT_{i,j}' = AT_{i',j'} + d$, where the distance between two neighboring cells, d, is either 1 or $\sqrt{2}$, from the definition of the $AT_{i,j}$ value of a cell $C_{i,j}$ in bucket LL_l. Thus, it can be concluded that $AT_{i,j} < c+1 \le AT_{i,j}'$ which is in contradiction to the assumption.

We then conclude that the $AT_{i,j}$ values of cells in the LL_{index} in step 2.3 will not be decreased and they are minimum values.

Theorem 1: The shortest path between any two different cells on a raster chart can be obtained from the algorithm if it exists.

Proof: If we determine a source cell, all of the cells surrounding the source cell, except the landmass, will be inserted into and, later, removed from the corresponding bucket exactly once during the computation. According to lemma 1, all of the $AT_{i,j}$ values of these cells will be minimized after

the computation. Thus, we can obtain the shortest path from the algorithm.

In the following, we prove the time complexity of this algorithm.

Lemma 2: Each cell in the electronic raster chart, except the land cell, will not be moved back to the bucket once it is removed.

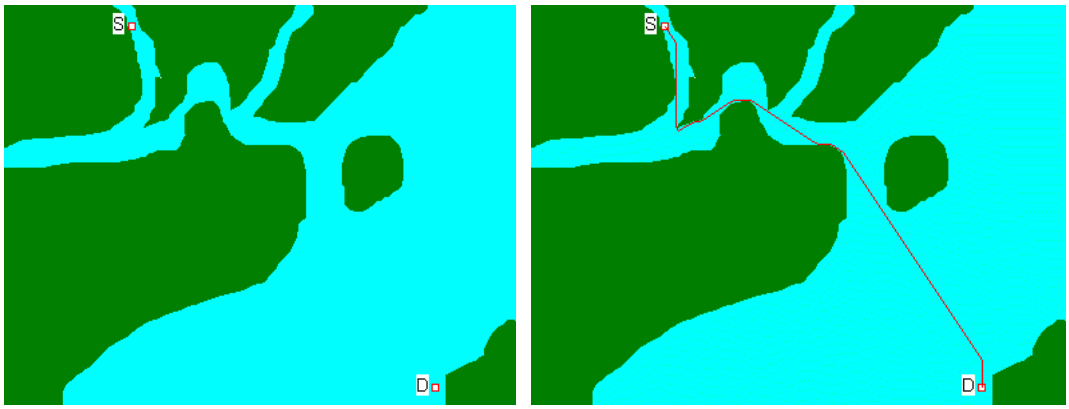
Proof: In this searching algorithm, the if and only if condition for the indices of cell $C_{i,j}$ to be moved into the bucket again is that the $AT_{i,j}$ value of the cell must be updated to a smaller value. According to lemma 1, each cell removed from the bucket has the minimum value. Therefore, the $AT_{i,j}$ value of the cell cannot be updated to a smaller value once the indices of cell $C_{i,j}$ are removed from the bucket. They will not be moved back into the bucket again.

Theorem 2: The shortest path algorithm has a time complexity of $O(N)$.

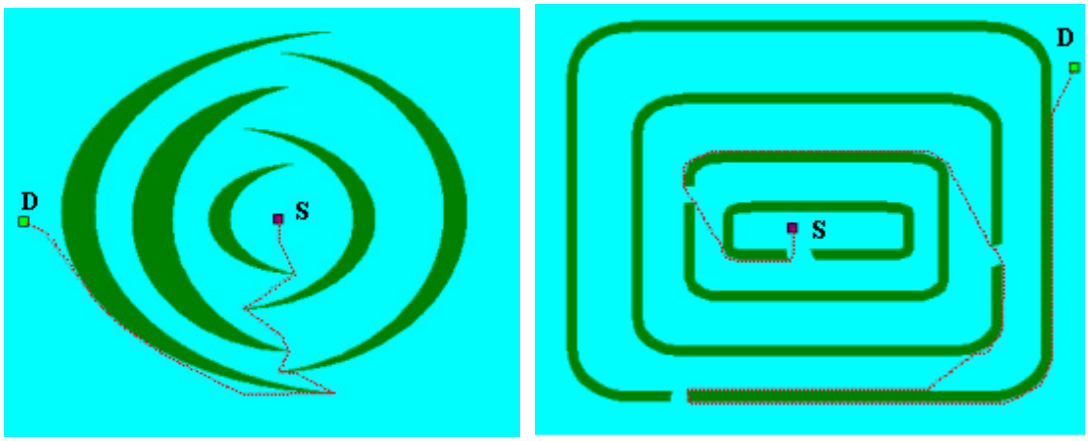
Proof: The steps 1, 3 and 4 of this searching algorithm have a time complexity of $O(N)$. According to lemma 2, once the indices of cell $C_{i,j}$ are removed from the bucket, it will not be moved into the bucket again. This means that the number of cells removed from the bucket are no greater than N during the process. Each cell has the time complexity of $O(1)$ to update its AT values for the 8 neighboring cells. We therefore know that the time complexity for step 2 is $O(N)$. It is concluded that the shortest path has the time complexity of $O(N)$ from steps 1 to 4.

A test program for one pair is illustrated in Figure 3.5. The Figure 9(a) identifies the source cell (expressed by S) and the destination cell (expressed by D). Figure 9(b) indicates the shortest path presented by a curve. Two more complicated

examples are depicted in Figure 9(c) and (d) as well.



(a) Determine the source cell and the destination cell. (b) Result from algorithm 1.



(c) Crescents pattern. (d) Maze pattern

Figure 9. Illustration of a test program for one pair.

3.5 THE SHORTEST PATH ALGORITHM FOR PLURAL PAIRS

The simplest method to solve the shortest path problem for plural pairs is to apply the algorithm introduced in section 3.4 to the individual vessel repeatedly. The difference between the shortest path algorithms for plural and single pairs is that the shortest paths for plural pairs must be obtained and displayed at the same time. Many vessels may have different speeds, thus an adjustment is made for the various speeds in the algorithm.

The *AT* values for the vessel shortest path are stored in the linked list as shown in Figure 4. We can modify the $AT_{i,j}$ value of the corresponding $C_{i,j}$ for the speed-varying system. The method for updating the $AT_{i,j}$ value for each node in the vessel shortest path is to divide the value of the *AT* field by the value of the speed field. Once the $AT_{i,j}$ value has been updated, the $AT_{i,j}$ value of each node in the vessel shortest path will be smaller if the vessel speed is larger than 1 cell/unit time. Otherwise, the $AT_{i,j}$ value will become larger if the vessel speed is less than 1 cell/unit time.

Algorithm 2: The shortest paths for plural pairs

Step 1: Obtain the shortest path for each vessel.

For each vessel Vessel_k, the source cell S_k and destination cell D_k to the shortest path algorithm are indicated. The corresponding shortest path is obtained, where $1 \le k \le q$, respectively.

Step 2: Update the *AT* values of the cells in the shortest path for each vessel.

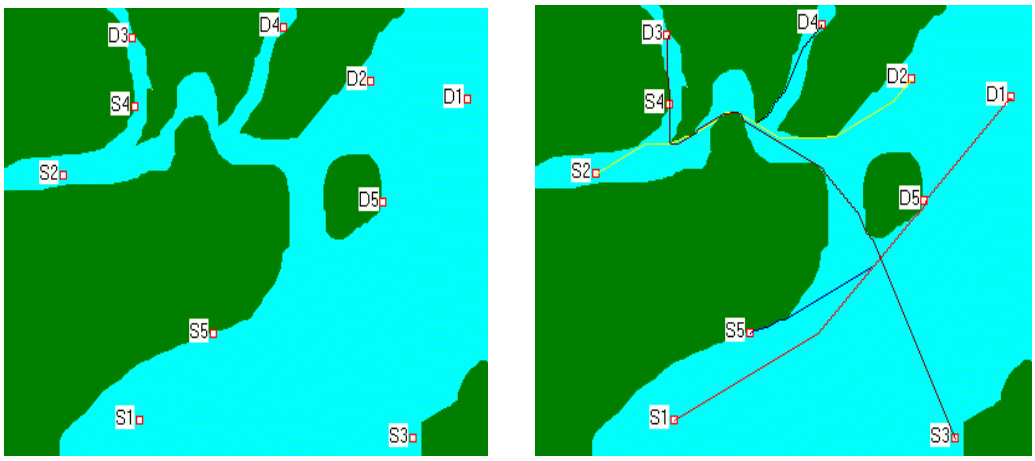
The original *AT* values divided by the speed are used to update the new *AT* values for the cells in the shortest path for each vessel.

Step 3: The collision detection.

If a potential collision is indicated, return an error message otherwise return “The shortest paths are obtained”.

END {Algorithm for the shortest paths of plural pairs}

An example of the shortest path algorithm for plural pairs is illustrated in Figure 10. The shortest path algorithm should be called q times for q vessels in executing of the shortest path algorithm for plural pairs, its time complexity thus is $O(qN)$.



(a) Identify the source and destination cells. (b) Result of algorithm 2.

Figure 10. Illustration of a test program for plural pairs.

4. CONCLUSION

This paper presented a planar shortest path algorithm on electronic maps based on the *nuclear fission chain reaction* scheme. In this study, we improved Lee's algorithm from a 4 cell-connected neighborhood to an 8 cell-connected neighborhood. The shortest path search, without crossing any barriers, was carried out along both oblique and rectilinear lines instead of rectilinear lines alone. Our new algorithm has $O(N)$ time and space complexities for a single pair shortest path search. When q vessels navigate in the same sea area, the algorithm has a time complexity of $O(qN)$. This idea can also be applied to marine search and rescue or sea interception systems performed on raster electronic charts. Especially for a camera-based monitoring system [19], the shortest path planning for robot motion on an image plane can be easily obtained by the proposed method.

In our future research, this method will be extended from a two-dimensional raster plane into a three-dimensional voxel space and from a static system into a dynamic (time varying) system by adding spatial and time parameters to the cell, respectively.

REFERENCE

[1] C. Y. Lee, "An Algorithm for Path Connection and Its Applications," *IRE Trans. Electron. Comput.*, Vol. EC-10, pp. 346-365, Sept. 1961.

[2] Frank Rubin, "The Lee Path Connection Algorithm," *IEEE Trans. on Comput.*, Vol. c-23, No. 9, pp. 907~914, Sept. 1974.

[3] J. H. Hoel, "Some Variations of Lee's Algorithm," *IEEE Trans. on Comput.*, Vol. c-25, No. 1, pp. 19~24, Jan. 1976.

[4] Gene Eu Jan, Ming-Bo Lin, and Yung-Yuan Chen, "Computerized Shortest Path Searching for Vessels," *Journal of Marine Science and Technology*, Vol. 5, No. 1, pp. 95-99, June 1997.

[5] D. Kirk, and L. Lim, "A Dual-mode Routing Algorithm for an Autonomous Roving Vehicle," *IEEE Transaction on Aerospace and Electronic Systems*, Vol. 6, No. 3, pp. 290-294, 1970.

[6] R. Larson, and G. Sadiq, "Facility locations with the Manhattan metric in the presence of barriers to travel," *Operations Research,* Vol. 31, pp. 652-669, 1983.

[7] J. Viegas, and P. Hansen, "Finding shortest paths in the plane in the presence of barriers to travel (for any l_p-norm)," *European Journal of Operational Research,* Vol. 20, pp. 373-381, 1985.

[8] Y. M. Chen, and P. Ramanan, "Euclidean Shortest Paths in the Presence of Obstacles," *Networks*, Vol. 21, pp. 257-265, 1991.

[9] K. Fagerholt, and A. Loktu, "Shortest Paths in the Presence of Obstacles: An Application to Ocean Shipping," *Journal of the Operational Research Society*,

Vol. 51, pp. 683-688, 2000.

[10] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik,* 1, pp. 269-271, 1959.

[11] S. N. Gewali, and G. T. Ioannis, “Path Planning in the Presence of Vertical Obstacles,” *IEEE Trans. on Robotics and Automation*, Vol. 6, No. 3, pp. 331-341, June. 1990.

[12] P. L. Lin, and Shyang Chang, “A Shortest Path Algorithm for a Nonrotating Object among Obstacles of Arbitrary Shapes,” *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 23, No. 3, May 1993.

[13] J. M. Ibarra-Zannatha, J. H. Sossa-Azuela, and H. Gonzalez-Hernandez, “A New Roadmap Approach to Automatic Path Planning for Mobile Robot Navigation”, *IEEE Intl. Conference on Systems, Man, and Cybernetics - PartA*, Vol. 3, pp. 2803 –2808, 1994.

[14] J. L. Díaz, de León S., and J. H. Sossa A., “Automatic Path Planning for a Mobile Robot Among Obstacles of Arbitrary Shape”, *IEEE Trans. on Systems, Man and Cybernetics - PartB*, Vol. 28, No. 3, June. 1998.

[15] M. Jern, “The Raster Graphics Approach in Mapping,” *Computers and Graphics*, Vol. 9, No. 4, pp. 373-381, 1985.

[16] J. H. Beattie, “The Future of Electronic Chart in Merchant Ships,” *The Journal of Navigation*,” Vol. 48, No. 3, pp. 335-348, 1995.

[17] John Dawson, “Digital Charting, Now and in the Future,” *The Journal of Navigation*, Vol. 52, No. 2, pp.251-255, 1997.

[18] W. R. Tobler, *Cellular Geography*. In S. Gale and G. Olsson, eds. *Philosophy*

in Geography, Dordrecht, Holland: D. Riedel Publishing Company, pp. 379-386.

[19] E. Kruse, and F. M. Wahl, "Camera-based Monitoring System for Mobile Robot Guidance," *Proceedings of The 1998 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Vol. 2, pp. 1248 –1253, Oct. 1998.