A 4-Geometry Maze Router and Its Application on Multiterminal Nets

GENE EU JAN
National Taipei University
KI-YIN CHANG
National Taiwan Ocean University
and
SU GAO and IAN PARBERRY
University of North Texas

The maze routing problem is to find an optimal path between a given pair of cells on a grid plane. Lee's algorithm and its variants, probably the most widely used maze routing method, fails to work in the 4-geometry of the grid plane. Our algorithm solves this problem by using a suitable data structure for uniform wave propagation in the 4-geometry, 8-geometry, etc. The algorithm guarantees finding an optimal path if it exists and has linear time and space complexities. Next, to solve the obstacle-avoiding rectilinear and 4-geometry Steiner tree problems, a heuristic algorithm is presented. The algorithm utilizes a cost accumulation scheme based on the maze router to determine the Torricelli vertices (points) for improving the quality of multiterminal nets. Our experimental results show that the algorithm works well in practice. Furthermore, using the 4-geometry router, path lengths can be significantly reduced up to 12% compared to those in the rectilinear router.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and routing*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Maze router, λ-geometry, cell map, Steiner tree

This work was supported by the National Science Council of Taiwan, ROC, under grant NSC-87-2211-E-019-004.

Authors' addresses: G. E. Jan, Department of Computer Science, National Taipei University, Sun Shia, Taipei County, 237, Taiwan; email: gejan@mail.ntpu.edu.tw; K.-Y. Chang, Department of Merchant Marine, National Taiwan Ocean University, Keelung, 202, Taiwan; email: b0170@mail.ntou.edu.tw; S. Gao, Department of Mathematics, University of North Texas, P. O. Box 311430, Denton, TX 76203; email: sgao@unt.edu; I. Parberry, Department of Computer Science, University of North Texas, P. O. Box 13886, Denton, TX 76203; email: ian@cs.unt.edu.

1. INTRODUCTION

The original objective of maze routing problems is to find a shortest path between two terminal cells on a rectangular grid of cells without crossing any obstacles. The first maze routing algorithm was presented by Lee [1961]. To this date, Lee's algorithm and its variations are still widely used in VLSI and PCB designs, in maze games, and in road map routing problems [Rubin 1974; Hoel 1976; Lin et al. 1990; Fawcett and Robinson 2000]. The popularity of Lee's algorithm lies in its simplicity and the guarantee it offers of finding a shortest path if one exists. However, Lee's algorithm is intrinsically based on the 2-*geometry* (also known as the *Manhattan geometry, rectilinear geometry,* etc.) of the grid plane. Each cell is considered to have only four neighbors, corresponding to at most four directions (left, right, up, and down) to move along an admissible path. This restriction is natural for some of the applications of the algorithm (e.g. VLSI designs), but unnecessary for most of the other situations.

To get around this restriction, many researchers took a graph-based approach to the problems arising in fields such as robot motion planning [Dí-az de León S. and Sossa A. 1998; Barraquand and Latombe 1991], computer-aided design [Xing and Kao 2002], and pattern recognition [Kimmel et al. 1995; Ogniewicz and Kubler 1995]. These graph-based approaches with suitable search structures solve the problems in substantially different ways. Some efficient implementations of Dijkstra's single-source shortest-path algorithm [Dijkstra 1959] have been proposed using Fibonacci heaps (F-heaps) to find the shortest path in the graph model [Ahuja et al. 1990; Henzinger et al. 1997]. For a grid with N cells, the implementation of Fibonacci heaps uses up to N F-heaps and the total complexity, including preprocessing, has not been improved to $O(N)$. Some other complicated data structures such as priority queues, hierarchical buckets, or multilevel buckets are required to avoid the sorting bottleneck even when the edge weights are bounded in most of these articles [Dinic 1978; Goldberg 2001; Thorup 1999]. In a computer-game application, the situation is naturally presented in a raster plane and the grid-based approach is easier to implement. Also, two advantages are obtained: independence from obstacles in the search process and freedom of preprocessing to construct a suitable search structure. In this article, we present a maze routing algorithm on the grid plane based on the 4-geometry. Our proposed algorithm provides simple and easy implementation when compared with other graph models, since it takes advantages of the nature of grid planes or raster planes and limited weights in λ-geometry (1 only in 2-geometry; 1 and $\sqrt{2}$ only in 4-geometry, 1; $\sqrt{2}$, $\sqrt{5}$ only in 8-geometry, and so on so forth). The algorithm is a directional improvement of Lee's algorithm, and therefore enjoys the two main advantages of Lee's algorithm: it is obstacle-independent and it guarantees finding the shortest path if one exists. Lee's algorithm fails for 4-geometry if different distances occur. An example will be given in Section 2 below. A straightforward attempt to improve Lee's algorithm by using equal cost wavefronts causes a substantial increase in time complexity [Fawcett and Robinson 2000]. Our algorithm works in a general context and uses a different data structure than that of Lee's algorithm. But in the case of 2-geometry, the two algorithms are the same. Also, our algorithm has the same time and space complexities of $O(N)$ as Lee's algorithm, where N is the number of cells in the grid plane. In fact, our algorithm uses at most twice the time as Lee's algorithm. Empirical data has shown that the overhead in running time is easily absorbed by the enhanced computing capabilities of modern computers.

It is worth mentioning that although we focus on the 4-geometry in this article because VLSI designers have no concern for the geometry value that is greater than 4, the algorithm works in more general situations. It can be easily adapted to handle general λ-geometry for $\lambda > 4$, and it can be used without substantial modification for higher dimensions. Thus considering 4-geometry is not an intrinsic restriction for us, although considering 2-geometry is an intrinsic restriction to applying Lee's algorithm.

Once the 4-geometry maze router has been introduced, the heuristic algorithm based on the maze router for the obstacle-avoiding Steiner tree problem will be presented. The main objective of this part of the article will be to implement both the 2-geometry and the 4-geometry maze routers in the multiterminal nets. The problem of optimally routing a multiterminal net in the presence of obstacles has received less attention. As a result, VLSI designers typically use a multiterminal variant of a maze routing algorithm and usually produce solutions that are far from optimal [Ganley and Cohoon, 1994]. Thus, one of the objectives of this study is to improve the quality of the multiterminal nets in the maze router. Ganley and Cohoon [1994] first presented an efficient model that allows computation of optimal obstacle-avoiding rectilinear (2-geometry) Steiner trees with three or four terminals. For nets with five or more terminals, they presented heuristic algorithms that work well in both theory and practice. Also, several researchers have successfully worked on the 1-Steiner tree problem [Georgakopolulos and Papadimitrious, 1987; Cieslik, 1991] and the heuristic 1-SMT (Steiner Minimal Trees) problem with a minimum spanning tree [Kahng and Robins, 1992; Cieslik, 1998]. Based on the concept of the 1-Steiner vertex (point) approach and the characteristics of the maze router, this article introduces a new cost accumulation scheme in the grid plane to obtain the Torricelli vertex, the minimum sum of the costs from all the terminals, for improving the quality of the multiterminal nets in global and local routings. The experimental data shows that the results are near optimal. Furthermore, experimental data has also shown that, by using the 4-geometry, path lengths can be reduced around 10–12% compared to those in the 2-geometry. This is a great improvement in many realistic applications by using the maze router. Finally, the need to design high-performance circuits has also called for the consideration of non-2-geometries (also known as *nonrectilinear* geometries) [Sherwani 1999; Teig 2002] in the field of VLSI design. This shows that there is a high potential in the applications of our 4-geometry router. An interesting observation is that the length reduction is quite marginal for higher geometries. Therefore it is effective enough to apply the 4-geometry for global routing problems. But the higher geometry is very useful for the other fields where more selective directions are needed to emulate the real routing situation.

The article is organized as follows. In Section 2 we give a description of our problem and formulate it in rigorous terms. In Section 3 the data structure is first described and defined. Then, the 4-geometry maze router is presented and the correctness and performance of the algorithm is proved and analyzed. The modification of the algorithm for the higher-geometry maze router is described as well. Some experimental results for the 4-geometry and 8-geometry routers are demonstrated. In Section 4 a heuristic algorithm based on the maze router is proposed for the multiterminal nets. Starting from a Torricelli vertex, the algorithm will search and connect a near optimal Steiner tree. After that, a cost accumulation scheme for subsets of three or four terminals among all terminals is included to refine the quality of the routing result. The experimental results show good solutions for obstacle-avoiding multi-terminal nets. The conclusions are summarized in Section 5.

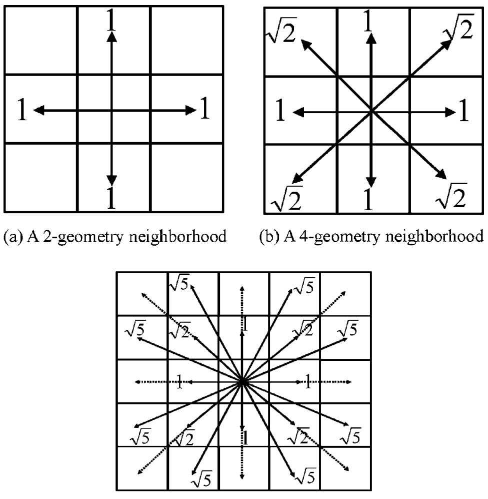
2. PROBLEM FORMULATION AND DESCRIPTION

Given an $m \times n$ rectangular grid E of N cells, a *cell map* is a function with domain E and a finite set of values that will be described in Section 3.1. The cell map indicates which cells constitute obstacles at the very least. As the algorithm unfolds, the cell map can also be used to indicate the intermediate status of the cells. A path on E is simply a sequence of cells on E. A path $p = (c_1, c_2, \ldots, c_n)$ is *admissible* if for each $i = 1, \ldots, n-1, c_i$ and c_{i+1} are neighboring cells and neither of them is an obstacle cell. It should also be given a monotone path cost function f, defined on the set of all paths in E. According to Lee, a function is *monotone* if for any path p and subpath q, $f(q) \leq f(p)$. For a given source cell S and a destination cell D, the *single-source routing problem* involves finding an admissible path p from S to D with the smallest possible value $f(p)$.

Note that the formulation of our problem is not in its most general form. This is done for the convenience of the reader and for simplicity in our presentation. In fact, the path cost function can in general be a vector function $F = (f_1, \ldots, f_r)$ with its values ordered lexicographically. Also our algorithm will generalize to three-dimensional grids without any difficulty.

In his original article [Lee 1961], Lee claimed to have solved the routing problem completely. But in fact Lee's algorithm does not work for the most general case [Rubin 1974]. In the rest of this section we give an example of a cell map for which Lee's algorithm does not yield the correct answer.

For an integer $\lambda \geq 2$, the λ-geometry allows edges with the angles of $i\pi/\lambda$, for all i, $\lambda = 2, 4, 8, 16$ and ∞ corresponding to rectilinear (90°), 45°, 22.5°, 11.25°, and Euclidean geometries, respectively. For a 2-geometry neighborhood, depicted in Figure 1(a), we are only interested in the cells above, below, to the left, and to the right of a cell, that is, all of the cells that are distance 1 unit from the center cell. The neighboring cells thus share an edge. If the four cells on the diagonal are included, we are working in a 4-geometry neighborhood, as depicted in Figure 1(b). In the 4-geometry, each cell has eight neighbors, and thus movements along the diagonal directions are possible. Note that Lee's algorithm can also be applied to the situation where each cell has eight neighbors. But, in order for Lee's algorithm to work, all eight neighboring cells must be equidistant from the center cell. In the 8-geometry neighborhood, each cell has 24 related neighbors. The distance of the 24-cell connected neighborhoods to the



(a) A 2-geometry neighborhood (b) A 4-geometry neighborhood

(c) An 8-geometry neighborhood

Fig. 1. Cell connection styles.

center cell is shown in Figure 1(c). There are 16 solid-line reachable neighbors that must be computed for each move. However, the eight dashed-line reachable neighbors do not need to be calculated since they can be extended by the next computation and will be computed in the next move. Note that each angle in the 8-geometry is not exactly 22.5°, half of 45°, because the angle is divided in the grid plane, not in a circular plane.

For the 16-geometry neighborhood, the running time for computation and condition statements is about twice that for the 8-geometry. However, the 16-geometry routing has twice the selective directions for searching a shortest path than that of the 8-geometry.

Lee's algorithm can be visualized as a wave propagation process. Putting the source cell in list L initializes the search. Two lists L and L_1 are defined to keep track of the cells on the wavefront (also called *frontier cells*) and their neighboring cells, respectively. After all the neighboring cells in L are included in L_1, the list L_1 is processed so that an expanded wavefront is found. Then any cell in L is deleted if all of its neighboring cells have been processed (updated) and L is updated by this new wavefront. The search is terminated if the destination cell is reached. This causes incorrect searching results if any single step is not equidistant. A simple example with a polygonal obstacle is shown in Figure 2. The result shows that the path p reaches the destination D first with the distance of 128.1 if Lee's algorithm is applied to the *4-geometry*. The path p' only reaches point A when path p reaches destination D. In fact, the shortest path from S to D is an extension of the path p' and has a shorter distance of 115. Thus Lee's algorithm no longer works for the 4-geometry.

In previous work, the radiation scheme was used for the conventional maze routing with diagonal connections to obtain the distances from the eight directional neighboring cells. The algorithm wasn't terminated until the AT values of the entire cells were the same in two consecutive steps if the obstacles of

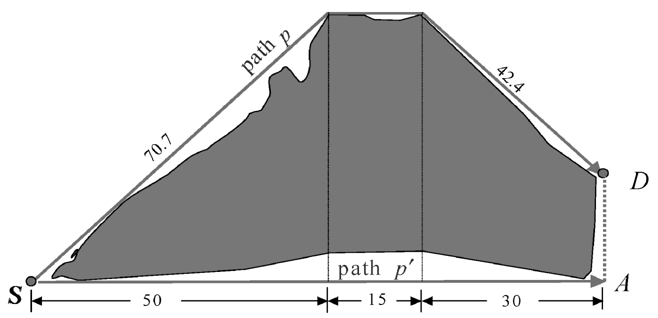


Fig. 2. Implementation of Lee's algorithm in the 4-geometry.

U-shape existed. Thus, the worse-case complexity was $O(N^2)$ [Jan et al. 1997]. Furthermore, a variant of Lee's router uses a time-ordered (cost-ordered) list to store the wavefront, rather than a simple list L. But the time complexity of their algorithm was described as polynomial in the number of edges [Fawcett and Robinson 2000]. In the next section, we solve the problem by introducing the concept of buckets. This data structure enables us to perform uniform propagation for arbitrary cost functions.

3. REQUIRED DATA STRUCTURE AND THE MAZE ROUTING ALGORITHM

The required data structure for this algorithm is mainly a cell map, and some linked lists and buckets. In the cell map, the cost function is represented by a so-called *AT*, or time of arrival, value. This follows a standard practice since the algorithm is first implemented for searching shortest path on the raster electronic map. The data structure and detail of the 4-geometry maze routing algorithm are introduced since it is of concern to most VLSI designers. The 8-geometry, 16-geometry, etc, have received less attention, but are beneficial for the other areas of application that will be discussed in the last subsection.

3.1 The Cell Map

The number of data fields in the cell map is flexible and depends on the requirements of individual applications. There are at least three parameters for cell storage in the 4-geometry, that is, F/O, *AT*, and *Vis*. The F/O (*Free space or Obstacle*) parameter distinguishes whether a cell is an obstacle, in which case the value is infinity, or a passable area, in which case the value is 1. The *AT* parameter stores the time (or other types of cost; in this article the *AT* value equals the distance divided by the velocity) 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 visited all its neighbors and update their *AT* values, and its initial Boolean value is *false*. For the initial cell condition, the black cells represent obstacles and the white cells represent passable areas. If the F/O parameter of any area has a finite value between 1 and infinity, we are working on the optimal path of various terrains. In an $m \times n$ grid plane, any cell $C_{i,j}$ has three parameters, $F/O_{i,j}$, $AT_{i,j}$, and $Vis_{i,j}$, where $0 \leq i \leq m-1, 0 \leq j \leq n-1$.

3.2 The Algorithm for 4-Geometry Maze Router

For an $m \times n$ grid of cells, the first step in the algorithm is to input 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 zero and the indices 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. The number of decimal digits in the irrational number, $\sqrt{2}$, is determined by the total number of cells on the grid plane. Therefore, the number of buckets can be reduced to three (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. In addition, a temporary list TL is applied to the algorithm to store the indices (i, j) of visited cells until all of the cells in the current bucket have been processed.

Since we have replaced the buckets β with three circular buckets, the three buckets are marked as LL_{index}, where the *index* is an integer variable and $0 \leq index \leq 2$. We also define a function Update_AT to support the structured program. The purpose of the function Update_AT is to update the $AT_{i,j}$ value 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_$AT$ ($(i, j)$, $new\_AT_{i,j}$):
Step 1: Update its $AT_{i,j}$ value of $C_{i,j}$ for an $m \times n$ grid plane.
If $new\_AT_{i,j}$ is less than $AT_{i,j}$, then $AT_{i,j} = new\_AT_{i,j}$.
END {Function of Update_$AT$}
```

The 4-geometry routing algorithm:

```
Initialization:
        For each cell $C_{i,j}(F/O_{i,j}, AT_{i,j}, Vis_{i,j})$ in an $m \times n$ grid plane, the initial $F/O_{i,j}$
        value is 1 if cell $C_{i,j}$ is in the free space or $\infty$ if it is in the obstacle. $AT_{i,j} = \infty$
        and $Vis_{i,j}$ = false for all cells, where $0 \leq i \leq m-1, 0 \leq j \leq n-1$. The initial
        value of index is 0.
Step 1: Input the coordinates of the source cell $S$ and the destination cell $D$. If $F/O_{i,j}$
        of the source cell is not equal to $\infty$, then update $AT_{i,j} = 0$, else return the
        error message "The source cell is in the obstacle" and terminate. If $F/O_{i,j}$ of the
        destination cell is equal to $\infty$, then return the error message "The destination
        cell is in the obstacle" and terminate.
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 list $TL$.
        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 $AT$ values of its neighboring
                  cells. Meanwhile, if the destination cell is in the current bucket, then
                  break step 2.
                  Step 2.3.1: Remove the indices of the first cell $C_{i,j}$ from the front end
                              of the $LL_{index}$ and update this cell's $Vis_{i,j}$ to true.
                  Step 2.3.2: Update the time of arrival ($AT_{i,j}$) of the 4-geometry neigh-
                              bors of cell $C_{i,j}$.
                              For each $C_{i,j}$'s neighbor $C_{i',j'}$, if the neighbor's $F/O = 1$
                              then call Update_$AT$ ($(i', j'), AT_{i,j} + 1$) for the 2-geometry
                              neighbors or call Update_$AT$ ($(i', j'), AT_{i,j} + \sqrt{2}$) for the
                              four diagonal neighbors.
```

```
            Step 2.3.3: Iterations
                        If $LL_{index}$ is not empty, then go back to step 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,
                        $index = (index+1) \bmod 3$.
    Step 2.5: Iterations
              If two consecutive buckets are not empty, then go back to step 2.3.
Step 3: Backtracking
        If the $AT_{i,j}$ value of the destination cell is infinity, return the error message:
        "There is not any path between this given pair". Otherwise backtrack the short-
        est path from the destination cell by selecting one of the 4-geometry neighbors
        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 4-geometry routing}
```

In the 4-geometry router, the wave propagation cannot stop right away when the destination is reached because another path from the same bucket or the next bucket may be a little bit shorter (by less than 1.0). However, if the destination cell is in the current executing bucket, then the algorithm stops since the AT value of the destination cell has been updated to a minimum value that will be proven in theorem 1. Figure 3 explains the detailed computation of the routing algorithm for a single pair. In this illustration, some diagonal paths that cross the tip of the obstacles should be considered as impassable to avoid the ambiguous situation. Figure 3(a) shows the initial grid plane with obstacles that have a source cell S and a destination cell D. After the execution of steps 1 and 2, the AT values of all the cells are obtained as shown in Figure 3(b). Then backtracking and reversing are implemented to obtain the shortest path, as shown in Figure 3(c).

If the 4-geometry routing algorithm is confined to 2-geometry, then it behaves the same as a Lee's algorithm. By comparing the result of Lee's algorithm shown in Figure 4 with that of the 4-geometry router shown in Figure 3(c), the distance from the source to the destination of latter is about 15% shorter.

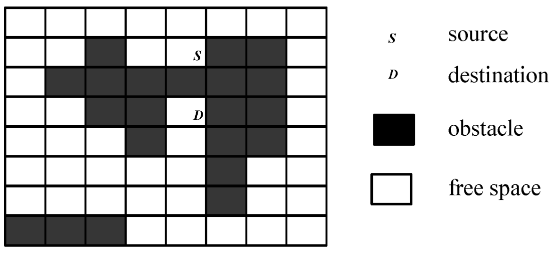
3.3 Performance Analysis of the 4-Geometry Maze Router

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

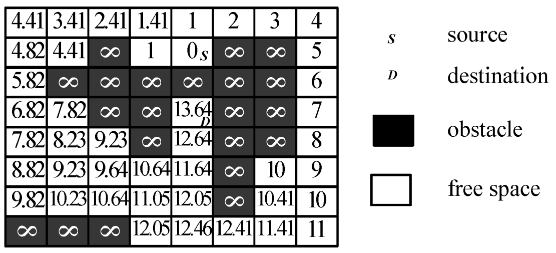
Observation 1. Once all the cells in the LL_{index} have updated the AT values of its neighboring cells and the LL_{index} is empty, all cells in the TL should be moved into either $LL_{(index+1) \bmod 3}$ or $LL_{(index+2) \bmod 3}$ according to their $AT_{i,j}$ values.

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

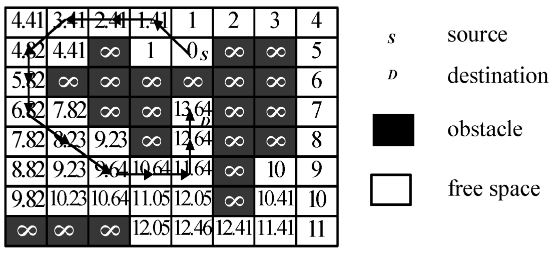
$$(\text{minimum } AT_{i,j} \text{value in the } LL_{index}) + 1$$



(a) The initial grid plane



(b) Execution result of steps 1 and 2



(c) Backtrack and reverse to obtain the shortest path

Fig. 3. Illustration of the 4-geometry routing algorithm on an 8 × 8 grid of cells.

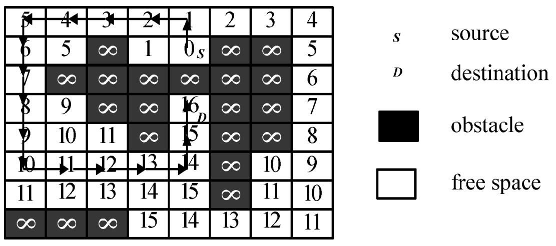


Fig. 4. Illustration of Lee's algorithm on the 8 × 8 grid of cells.

and the maximum $AT_{i,j}$ value of cells in the TL would be no greater than

$$(\text{maximum } AT_{i,j} \text{ value in the } LL_{index}) + \sqrt{2}.$$

The LL_{index} is empty when step 2.3 is completed. After the computation of step 2.4.1, all of the cells in the temporary list 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 $AT_{i,j}$ values of cells in the LL_{index} are minimal in step* 2.3.

PROOF. This lemma is proven by contradiction. For the $AT_{i,j}$ value of any cell $C_{i,j}$ in the current bucket LL_c, where $0 < c \leq AT_{\max}$, 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 $AT_{i,j}$ value of the cell $C_{i,j}$, should be replaced by $AT'_{i,j}$ if the cell $C_{i',j'}$ is in one of the previous buckets LL_l where $0 \leq l \leq c - 1$ and LL_c is the current bucket. If the cell $C_{i',j'}$ is in the current or one of the higher buckets LL_l, where $c \leq l \leq AT_{\max}$, we know that $c \leq AT_{i',j'}$ and $c \leq AT_{i,j} < c + 1$. Also $AT'_{i,j} = AT_{i',j'} + dist$, where the distance between two neighboring cells, *dist*, is either 1 or $\sqrt{2}$, from the definition of the $AT_{i,j}$ value of a cell $C_{i,j}$ in the bucket LL_l. Thus, it can be concluded that $AT_{i,j} < c + 1 \leq AT'_{i,j}$, 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 are minimal. □

THEOREM 1. *The shortest path between any two different cells on a grid plane can be obtained from the algorithm if it exists.*

PROOF. If a source cell is determined, all of the cells surrounding the source cell, except the obstacles, will be moved 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 are minimal after the computation. Thus, we can obtain the shortest path from the algorithm. □

LEMMA 2. *Each cell in the grid plane, except the obstacle cell, will not be moved back to the bucket once it is removed.*

PROOF. In this searching algorithm, the indices of cell $C_{i,j}$ are moved into the bucket again if and only if the $AT_{i,j}$ value of the cell is 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. □

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

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

PROOF. 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 the AT values of its eight 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. □

The complexity of this algorithm would increase by $O(U/L)$ only if U/L can be normalized, where U and L are the upper and lower bounds for edge weights, since we would need at least $(U/L)N$ number of buckets in that case.

3.4 The Higher-Geometry Maze Router

For the 8-geometry maze router, the 4-geometry router can be modified by simply increasing the number of circular buckets to four and some conditional statements, for the $\sqrt{5}$ step move. Meanwhile, a *Dir* parameter in the cell map is required to keep track of which predecessor causes the minimum *AT* value. The 8-geometry uses at most twice the time as the 4-geometry. Thus, for the 16-geometry, 32-geometry, and other higher-geometry maze routers, it is merely required to increase the number of circular buckets and conditional statements. The running time is $O(\lambda N)$ for the λ-geometry algorithm. The performance analysis of the higher geometry routers is neglected since it is similar to the 4-geometry case.

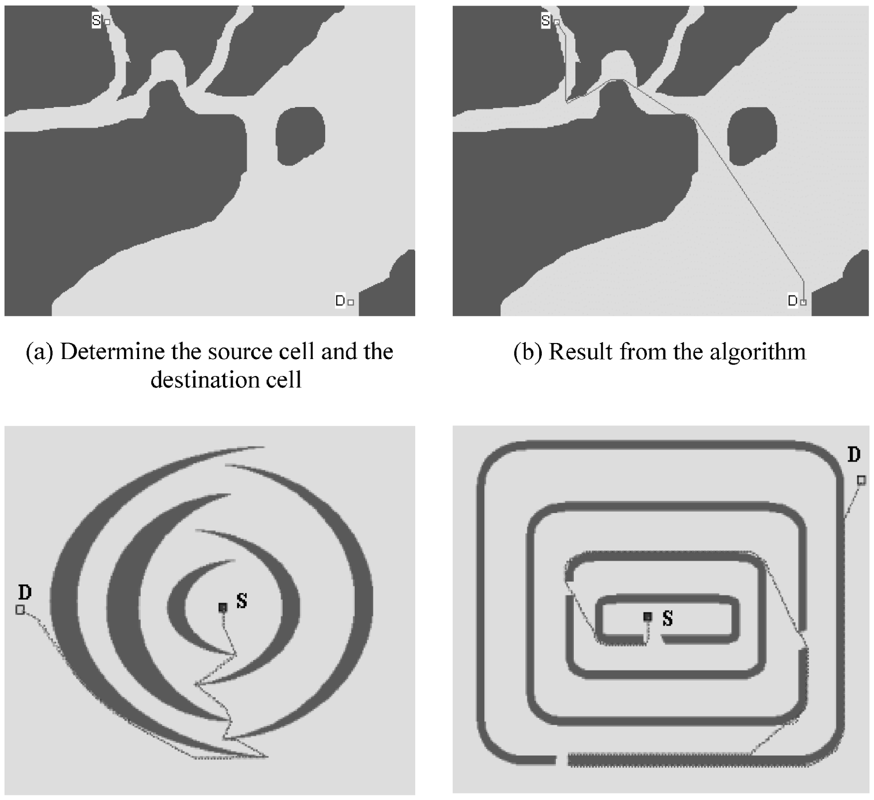
Furthermore, the F/O parameter is introduced for various terrains in which the velocity of the object for those cells in the specific terrains is divided by the value of its F/O parameter. The uniform wave propagation for various terrains can be achieved by increasing a certain number of buckets. As a whole, the algorithm is capable of finding the optimal path in the grid plane among various terrains. If there is only one terrain in the grid plane, the optimal path is also called the *shortest path*.

3.5 Experimental Results

An example of the maze routing on a raster electronic map is illustrated in Figure 5. The source cell S and destination cell D are marked in Figure 5(a). In a 400×300 pixel (considered as cells) raster electronic map, finding the shortest path by the 4-geometry maze router between a source cell and a destination cell takes only a fraction of a second for the Pentium III PC. Figure 5(b) indicates the shortest path presented by a curve. Figures 5(c) and 5(d) are two examples depicted by implementing the 8-geometry router, in which each 45° angle in the 4-geometry has been divided into angles of 26.4° and 18.6°. It is interesting to compare our algorithm with the popular A* algorithm [Hart et al. 1968] used in computer games, but note that the A* algorithm can be trapped in some mazes, whereas our algorithms guarantee finding the shortest path if one exists.

4. ROUTING OF MULTITERMINAL NETS

The problem of finding the minimal-length wire routing for multiterminal nets belongs to the class of Steiner tree problems. The *Steiner tree problem* is a minimum interconnection problem. The problem can also be applied in the geometric realm; the most common variants are the Euclidean Steiner tree and the rectilinear Steiner tree problems. In additional to the Euclidean and rectilinear metrics, octagonal metrics (4-geometry) has been considered. Here, the input is a set of vertices in the plane that are the terminals, and the goal is to obtain a tree of minimum length (in the appropriate metric) that connects all the terminals. Up to now, many exact, heuristic, and genetic algorithms have been proposed



(a) Determine the source cell and the destination cell

(b) Result from the algorithm

(c) Crescents pattern

(d) Maze pattern

Fig. 5. Illustrations of the maze routing on the raster electronic map.

for constructing these trees [Lee et al. 1976; Hwang 1979; Smith et al. 1981; Dreyer and Overton 1998]. However, only some of these algorithms solve the Steiner tree in a plane with obstacles [Winter, 1993; Ganley and Cohoon, 1994]. The problem of constructing Steiner minimal trees in the Euclidean plane without obstacles is NP-hard. When obstacles are included, the obstacle-avoiding Steiner minimal-trees problem is at least NP-hard as well [Cieslik, 1998]. The main objective of this section is to develop a heuristic algorithm based on the maze routers to construct Steiner trees in 2-geometry and 4-geometry. The algorithm introduces a (global) Torricelli vertex to construct the obstacle-avoiding Steiner trees based on the cost accumulation of the cells. After that, a cost accumulation scheme is also used to improve the Steiner tree locally. The algorithm is presented for the rectilinear case since the exact solution of rectilinear Steiner trees for small instances can be obtained without any difficulty.

4.1 Definition of the Torricelli Vertex

For multiterminal nets problems, the maze routing approach is one of the basic algorithms. The traditional maze router connects the first two terminals, and then the entire connected path is the target for the wave propagation from the third terminal, etc. Thus, all the r terminals of $Z = (Z_1, \ldots, Z_r)$ are considered as the sources for the wave propagation to the other terminals at least once. The quality of its result strongly depends on the ordering of the terminals since it is difficult to determine the optimal terminal ordering. In order to improve the routing quality of multiterminal nets, a Torricelli vertex (point), a vertex that has the minimum total cost to all terminals, is introduced for further routing. The definition of this vertex can be considered as the following geometric problem:

Given Z, where terminal i is at location (x_i, y_i), find a vertex (x_c, y_c) such that $D_c = \sum_{1\leq i\leq r} d_i$ is minimum, where, d_i is the distance from the ith vertex (x_i, y_i) to (x_c, y_c), and r is the total number of vertices. The vertex (x_c, y_c) has the minimum total cost connecting to those r terminals including superposition connection. The cost (length) of the near optimal Steiner tree is no more than the value of the Torricelli vertex since the superposition (overlap) path will be removed after the routing process. If the maze router uniformly propagates from each source vertex to the remaining vertices, then the cost (time of arrival) of all vertices in the cell map is specified. The cells with equal cost in the grid plane can be considered as a level curve, which is similar to the sea-level curve in a geographical map. A solution of this problem is to obtain a vertex that minimizes the sum of the cost from all of the r terminals. Thus, the minimization of the above problem can be formulated as the following algebraic program:

Minimize $\sum_{1\leq i\leq r} d_i$
subject to $(x_c - x_i)^2 + (y_c - y_i)^2 \leq d_i^2, i = 1, \ldots, r,$

where $d_i \geq 0$ and r is the total number of terminals.

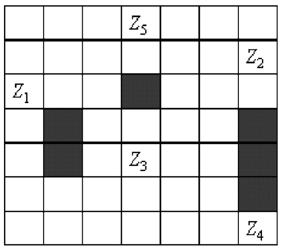
This is Fermat's problem in a Euclidean plane [Brimberg 1995], which is the local version of Steiner's problem. If the r terminals are not collinear in a Banach-Minkowski space $M_d(B)$ and if B is a strictly convex unit ball, then the Torricelli vertex (Steiner vertex) is a single vertex. A Banach-Minkowski space is uniquely defined by a d-dimensional affined space A_d and convex and compact body B. Usually, a (finitely or infinitely dimensional) linear space, which is complete with regard to its given norm, is called *Banach space*. This is a natural idea to construct a 1-Steiner (Torricelli) as an approximation of an SMT (Steiner minimal tree). Instead of using the iterative Weiszfeld's algorithm [Kuhn 1973] in the above algebraic program, our algorithm introduces a cost accumulation scheme to obtain the Torricelli vertex.

4.2 An Algorithm for Near Optimal Rectilinear Steiner Trees

For rectilinear Steiner trees (RSTs), Fermat's problem can be described as follows:

Minimize $\sum_{1\leq i\leq r} |x_c - x_i| + |y_c - y_i|$, where (x_c, y_c) has the minimum total rectilinear cost to all terminals.

The goal of our algorithm is to find a near optimal Steiner tree starting from (x_c, y_c) to connect all the terminals. The minimum rectilinear Steiner tree



(a) Initial state

31	28	25	24	23	24	27
28	25	22	21	18	21	24
27	24	21	■	21	22	25
30	■	22	21	20	21	■
31	■	23	22	21	22	■
32	29	26	25	24	25	■
35	32	29	28	27	28	31

(b) Cost accumulation table from Z_1 to Z_5

Fig. 6. Example of the cost accumulation by using the maze routing algorithm.

problem in the plane is defined as follows: given a set of r terminals in an $m \times n$ grid plane of N vertices, a set S_t of Steiner vertices such that the spanning tree over $Z \cup S_t$ has minimum cost is found.

The cost accumulation scheme presented in this article is to obtain the Torricelli vertex for further routing and local refinement of multiterminal nets. The way to find the set of Torricelli vertices is still unknown, but a single global Torricelli vertex can be obtained by the cost accumulation of the r terminals in this algorithm. For instance, there are five terminals that are designed to connect to an RST, as shown in Figure 6(a). Thus, the five cost tables from Z_1 to Z_5 are generated individually. The cost accumulation table is the sum of costs in the five tables, as shown in Figure 6(b), where the Torricelli vertex is the vertex with the minimum value in the cost accumulation table. Note that the cost accumulation scheme uses extra memory space to reduce the time complexity.

Once the cost accumulation scheme is applied in the grid plane, the algorithm starts from the Torricelli vertex to approximate a near optimal Steiner tree. First of all, we can obtain the nearest terminal Z_i from the Torricelli vertex, and then the first critical path is connected from Z_i to its nearest terminal Z_j, where Z_i and Z_j belong to Z. After that, the entire connected paths are considered as the targets to the remaining terminals. Based on the sum of costs in the remaining terminal tables, the next critical path is a shortest path between a nearest terminal and the existing (entire) connected paths, and the new entire connected paths are then the combination of this new critical path and the previous entire connected paths. Continue these steps until all terminals are connected. Also, a critical vertex is defined as the connection vertex between the shortest path and the entire connected paths.

For the local refinement, the same cost accumulation scheme is applied locally if the localized subset is determined. The local refinement selects subsets of three or four connected vertices by using the method of the Steiner visibility region [Winter 1993]. In order to search and locate which vertices are most likely to appear near each other in the tree, the beam of light originating from the global Torricelli's vertex is rotated all the way around in the grid plane. The local Torricelli (Steiner) vertex connects these three or four vertices and the Steiner tree replaces the original path if the rerouted Steiner tree has a smaller length (cost) than the original one. Also, it should be noted that the cost accumulation scheme starts from a Torricelli vertex and may induce several Steiner vertices by using the maze router to connect the multiterminal nets. The algorithm can be summarized in the following steps:

Heuristic algorithm:

Step 1: Individual costs
For each terminal in Z, a wave propagates to the remaining vertices in the cell map and stores its costs in the corresponding cost table. (These r cost tables generated by the r terminals are stored and used for the rest steps of our algorithm.)

Step 2: Cost accumulation
A cost accumulation table is generated based on the sum of the r cost tables. The vertex with minimum total cost in the cost accumulation table is considered as a Torricelli vertex for further constructing the SMT.

Step 3: Initial critical path
Obtain the nearest terminal Z_i from the Torricelli vertex by an index search in the r tables to find the minimum cost between all the r terminals and the Torricelli vertex. The first critical path of multiterminal nets is obtained from the Z_i that backtracks to its nearest terminal Z_j, where Z_i and Z_j belong to Z.

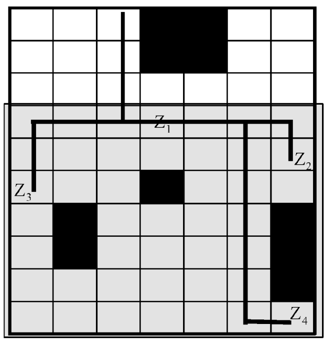
Step 4: Iterations
The entire connected paths are considered as the target vertices to the remaining terminals. Once the critical vertex is determined, backtrack to the nearest terminal to obtain the shortest path and then insert it into the entire connected paths. For the purpose of shortening the sum of the lengths in the remaining critical paths, choose the optimal one according to the minimum sum of costs in the remaining terminal tables whenever two or more shortest paths exist. Both the critical vertex and its nearest terminal can be found by an index search in the $r - 1$ tables with sentinel data structure. Repeat this step until all of the terminals are connected. The final obtained connected paths are the near optimal Steiner tree for the given r terminals.

Step 5: Local refinement
Identify subsets of three or four terminals of Z, which are most likely to appear near each other in the tree by the method of the Steiner visibility region. The local Torricelli vertices are obtained by the cost accumulation scheme of these subsets. From these vertices, the cost accumulation scheme, as in steps 3 and 4, is applied to obtain the local minimal Steiner tree as shown in Figure 7, in which the particular local Steiner tree has four terminals. Compare the rerouted path with the local tree (connected paths) in this subset and replace the local tree if the new tree has a smaller length. Repeat this process for the whole Steiner tree.

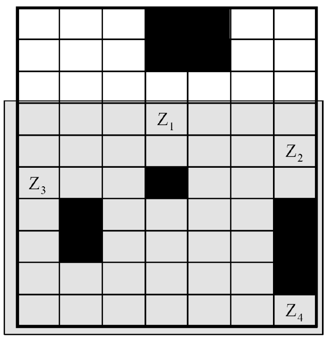
The space and time complexities of the proposed heuristic algorithm are easily analyzed and can be obtained as $O(rN)$ and $O(r^2N)$, respectively.

4.3 Exact Algorithm of Minimum Rectilinear Steiner Trees

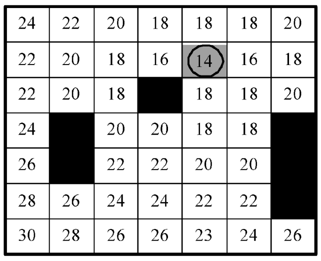
Corresponding to multiterminal routing, the rectilinear Steiner minimum tree problem cannot be solved exactly by any polynomial time algorithms. Nonetheless, exponential time algorithms have been devised to solve the SMT problem exactly for small instances in a reasonable time. Here, an exact algorithm is computed to compare the heuristic result. The exact algorithm for the SMT problem is an exhausted searching method. Once the numbers and locations of the terminals and obstacles are given, the algorithm runs all the possible combination for free vertices if all of the terminals can be connected. To reduce



(a) Localized from a part of rectilinear Steiner tree

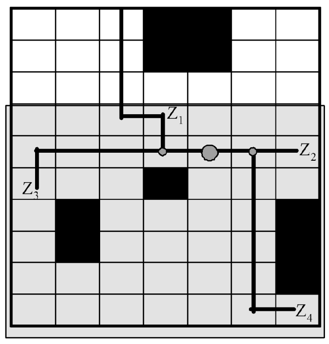


(b) Rip-up



24	22	20	18	18	18	20
22	20	18	16	14	16	18
22	20	18		18	18	20
24		20	20	18	18	
26		22	22	20	20	
28	26	24	24	22	22	
30	28	26	26	23	24	26

(c) Obtain the Torricelli point (Steiner point) from the cost accumulation table



(d) Result of the cost accumulation scheme

Fig. 7. An example of the cost accumulation scheme in step 5.

the overall computation, an upper bound is defined as the length obtained from the heuristic algorithm and the number of terminals minus 1 is defined as a lower bound.

4.4 Experimental Results

We have analyzed the performance of this algorithm for the examples of multi-terminal nets. The given parameters within a square region (e.g., 10×10, and some of 20×20) are the samples that include obstacles and specified r terminals. The locations of terminals and obstacles are randomly generated. For the given numbers of terminals and obstacles (also called a *set of samples*), if the set of samples runs all of its combination in the grid plane for more than 5 h then it is discarded. According to the density of the terminals (nodes) and obstacles, we have 42,890 effective samples, as shown in Figure 8(a), that run both heuristic and exact algorithms. The distribution of length differences between the heuristic and exact solutions is shown in Figure 8(b). From the statistical criteria, we first calculate the difference of each sample by$X_i = length_{heuristic} - length_{exact}$, where $0 \leq i < n$ and n is the number of the samples. After that, we obtain the mean, $\overline{X} = \sum_{i=0}^{n-1} X_i/n$, and calculate the confidence interval *CI* by the formulation of the statistical criteria $CI = \pm 1.96(\overline{X}/\sqrt{n})$. Next, we calculate 95% of

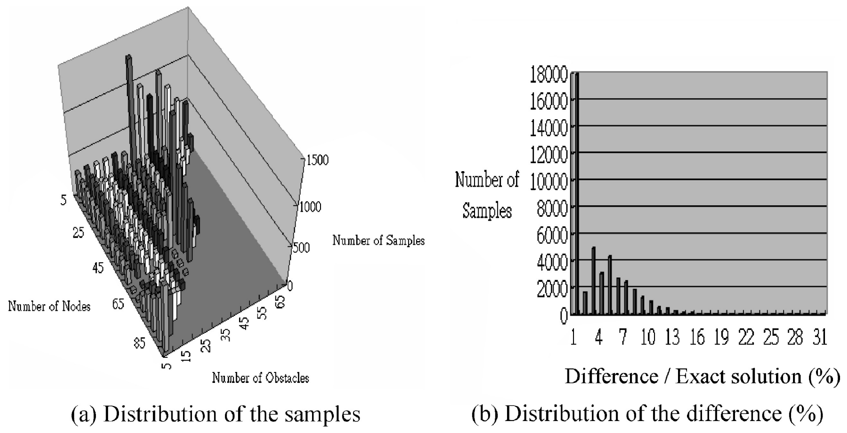


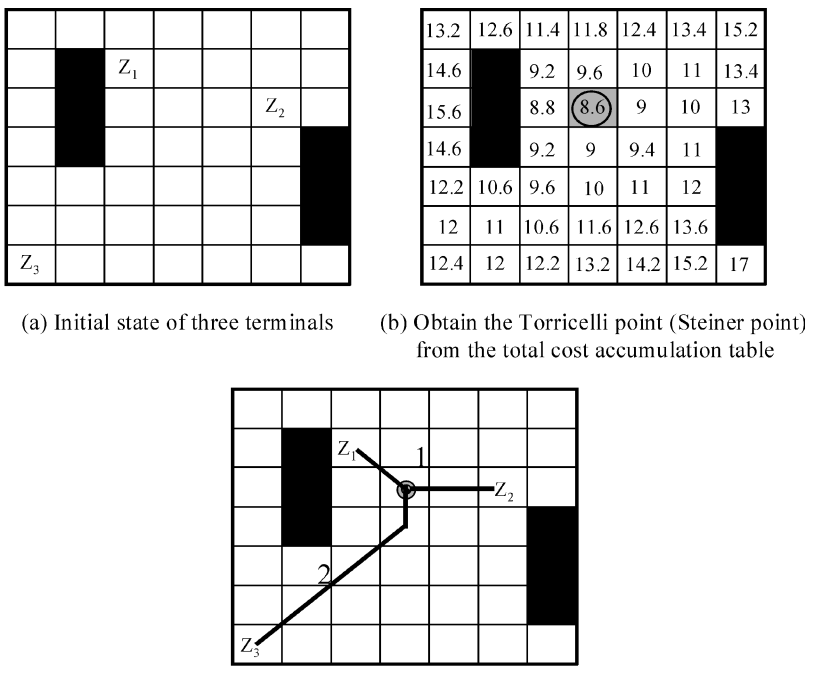
Fig. 8. Distribution of the samples and their differences with exact solutions.

$CI/\overline{X} = 0.0195 \leq 0.025$ to determine whether the sample is reliable. The standard deviation is 1.34, and that means the standard deviation value between the heuristic and exact algorithms is between 0–1.34 units of length. Note that the heuristic solutions have a larger difference if the number of the terminals is large, since it is more like a k-Steiner tree problem. We present the empirical evidence that this algorithm implies a good solution for multiterminal nets in the obstacle-avoiding Steiner tree problem. For example, the worst case is bounded by 2.52 units of length for a 10×10 grid plane.

4.5 4-Geometry Steiner Trees

Recently, nonrectilinear geometry has gained ground because of enhanced computing capabilities and the need for the design of high-performance circuits. Our heuristic algorithm can be extended to 4-geometry since it was designed to find near optimal Steiner trees in the grid plane. The procedures of constructing the 4-geometry Steiner tree are similar to those for the rectilinear Steiner tree except that the diagonal direction is included in the backtracking step.

After the Torricelli point is determined from the cost accumulation table, as shown in Figure 9, we can obtain its nearest terminal, Z_1. The constructing of critical path 1 backtracks from Z_1 to its nearest terminal, Z_2. The constructing of critical path 2 backtracks from the vertices in path 1 to Z_3, etc. An example in a 20×20 grid plane that has 12 connecting terminals with components (obstacles) is presented in Figure 10 to demonstrate the difference between the rectilinear and 4-geometry Steiner trees. Figure 10(a) is the experimental result, showing the length of the rectilinear tree to be 66 units, where the squares filled with black represent the circuit components considered as the obstacles and the circles represent the connecting terminals. Also, this experimental result has the same length as the exact solution, taking 0.12 s on a Pentium III PC with 256 MB of memory. If the 4-geometry maze router is applied to this sample, the length is 58.87 units, as shown in Figure 10(b), and is nearly reduced by 11% compared to the rectilinear case. It is time-consuming for the 4-geometry to get the exact solution; only some small nets (10×10) were tested, and the

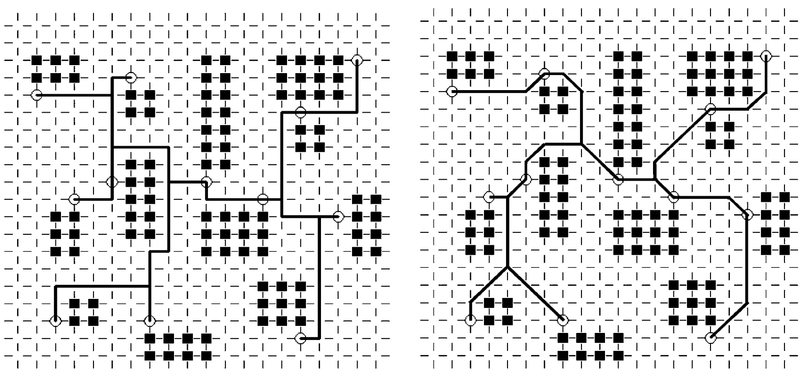


13.2	12.6	11.4	11.8	12.4	13.4	15.2
14.6	■	9.2	9.6	10	11	13.4
15.6	■	8.8	8.6	9	10	13
14.6	■	9.2	9	9.4	11	■
12.2	10.6	9.6	10	11	12	■
12	11	10.6	11.6	12.6	13.6	■
12.4	12	12.2	13.2	14.2	15.2	17

(a) Initial state of three terminals (b) Obtain the Torricelli point (Steiner point) from the total cost accumulation table

(c) Order of path construction and result of a 4-geometry Steiner tree

Fig. 9. Example of constructing a 4-geometry Steiner tree.



(a) A rectilinear Steiner tree (b) A 4-geometry Steiner tree

Fig. 10. Illustration of rectilinear and 4-geometry Steiner trees.

results showed the length difference between the heuristic and exact solutions to be around 0.67% in the 1000 test samples. The worst case had 2.8 units of length difference in one sample.

However, the difference in length reduction was quite marginal for the 4-geometry and the 8-geometry routings. Also, the terminal ordering problem may increase the length slightly in the higher-geometry routings compared to the 4-geometry routing. Thus, it is effective enough to run the 4-geometry routing for most of the SMT problems with obstacles. The main difference between our router and other 4-geometry routers is that our algorithm can work on various terrains and also work on a geometry value higher than 4 (i.e., 8-geometry, 16-geometry, etc.), which is useful for other research areas.

5. CONCLUSIONS

This article presents a 4-geometry maze routing algorithm and its application to Steiner tree problems. In this study, we improved the traditional maze router by properly using the data structure of buckets, a simple temporary liked-list, *TL*, cell matrices and indices (i, j) (sequentially allocated the memory in constant time) with linear space to avoid any sorting or presorting. In regard to 4-geometry, we were able to significantly shorten the optimal path between a source cell and a destination cell. Our new algorithm has both $O(N)$ time and space complexities since we did not sort or presort the edge weights at all. Without any difficulty, the algorithm is capable of being extended to 8-geometry, 16-geometry, etc., for use in other research areas such as the path planning of robot motion, raster charts, or nondistorted digital maps, etc. For the various terrains problem, the algorithm is able to find an optimal path between a given pair of cells by modifying the number of required buckets.

For the multiterminals net, the cost accumulation scheme was applied to the grid-based Steiner tree problems, in which the Torricelli vertices are located, to improve the routing. The experimental results showed that the algorithm works well in the RST problems. In addition, using the 4-geometry routing, path lengths can be reduced around 10–12% compared to those in the 2-geometry routing. But if the geometry value is higher than 4, the gain for the obstacle-avoiding Steiner tree is quite marginal.

The proposed method can be extended from two-dimensional grid planes to three-dimensional volumes and from static systems to dynamic (time-varying) systems by adding spatial and time parameters, respectively. It also can be applied to multilayer global routing problems.

REFERENCES

AHUJA, R. K., MEHLHORN, K., ORLIN, J. B., AND TARJAN, R. E. 1990. Faster algorithms for the shortest path problem. *J. Assoc. Comput. Mach. 37*, 2 (April), 213–223.

BARRAQUAND, J. AND LATOMBE, J. C. 1991. Robot motion planning: A distributed representation approach. *Int. J. Robotics Res. 10*, 6 (Dec.), 628–649.

BRIMBERG, J. 1995. The Fermat-Weber location problem revisited.*Math. Programm. B 71*, 1, 71–76.

CIESLIK, D. 1991. The 1-Steiner-minimal-tree problem in Minkowski-spaces, *Optimization 22*, 2, 291–296.

CIESLIK, D. 1998. *Steiner Minimal Trees*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

DÍ-AZ DE LEÓN S., J. L., AND SOSSA A., J. H. 1998. Automatic path planning for a mobile robot among obstacles of arbitrary shape. *IEEE Trans. Syst. Man. Cybernet. B. 28*, 3 (June), 467–472.

DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numer. Math. 1*, 269–271.

DINIC, E. A. 1978. Economical algorithms for finding shortest paths in a network, In *Transportation Modeling Systems*, Y. S. Popkov and B. L. Shmulyian, Eds. Institute for System Studies, Moscow, Russia, 36–44.

DREYER, D. R. AND OVERTON, M. L. 1998. Two heuristics for the Euclidean Steiner tree problem. *J. Glob. Opt. 13*, 1, 95–106.

FAWCETT, J. AND ROBINSON, P. 2000. Adaptive routing for road traffic. *IEEE Comput. Graph. Appl. 20*, 3 (May/June), 46–53.

GANLEY, J. L. AND COHOON, J. P. 1994. Routing a multi-terminal critical net: Steiner tree construction in the presence of obstacles. *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1. 113–116.

GEORGAKOPOLULOS, G. AND PAPADIMITRIOUS, C. H. 1987. The 1-Steiner tree problem. *J. Algorith. 8*, 1, 122–130.

GOLDBERG, A. V. 2001. Simple shortest path algorithm with linear average time. *Proceedings of the European Symposium on Algorithms (ESA)*. Lecture Notes in Computer Science, Vol. 2161. Springer-Verlag, Berlin Germeny, 230–241.

HART, P. E., NILSSON, N. J., and RAPHAEL, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybernet. 4*, 2, 100–107.

HENZINGER, M. R., KLEIN, P., RAO, S., AND SUBRAMANIAN, S. 1997. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci. 55*, 1 (Aug.), 3–23.

HOEL, J. H. 1976. Some variations of Lee's algorithm. *IEEE Trans. Comput. C-25*, 1 (Jan.), 19–24.

HWANG, F. K. 1979. An $O(n \log n)$ algorithm for suboptimal rectilinear Steiner trees. *IEEE Trans. Circ. Syst. CAS-26*, 75–77.

JAN, G. E., LIN, M. B., AND CHEN, Y. Y. 1997. Computerized shortest path searching for vessels. *J. Marine Sci. Tech. 5*, 1, 95–99.

KAHNG, A. AND ROBINS, G. 1992. A new class of iterative Steiner tree heuristics with good performance.*IEEE Trans. Comput.-Aided Des. Integrat. Circ. Syst. 11*, 7, 893–902.

KIMMEL, R., AMIR, A., AND BRUCKSTEIN, A. M. 1995. Finding shortest paths on surfaces using level sets propagation. *IEEE Trans. Patt. Anal. Mach. Intell. 17*, 6 (June), 635–640.

Kuhn, H. W. 1973. A note on Fermt's problem. *Math. Programm. 4*, 98–107.

LEE, C. Y. 1961. An algorithm for path connections and its applications. *IRE Trans. Electron. Comput. EC-10*, Sept., 346–365.

LEE, J. H., BOSE, N. K., AND HWANG, F. K. 1976. Use of Steiner's problem in suboptimal routing in rectilinear metric. *IEEE Trans. Circ. Syst. CAS-23*, 470–476.

LIN, Y. L., HSU, Y. C., AND TSAI, F. S. 1990. Hybrid routing. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst. 9*, 2 (Feb.), 151–157.

OGNIEWICZ, R. L. AND KUBLER, O. 1995. Voronoi tessellation of points with integer coordinates: Time-efficient implementation and online edge-list generation. *Patt. Recogn. 28*, 12 (Dec.), 1839–1844.

RUBIN, F. 1974. The Lee path connection algorithm. *IEEE Trans. Comput. C-23*, 9 (Sept.), 907–914.

SHERWANI, N. A. 1999. *Algorithms for VLSI Physical Design Automation*, 3rd. ed. Kulwer Academic Publishers, Boston, MA, 260–279.

SMITH, J. M.-G., LEE, D. T., AND LIEBMAN, J. S. 1981. An $O(n \log n)$ heuristic for Steiner minimal tree problems on the Euclidean metric. *Networks 11*, 1, 23–39.

TEIG, S. L. 2002. The X architecture: Not your father's diagonal wiring. In *Proceedings of the 2002 International Workshop on System-Level Interconnect Prediction* (San Diego, CA, Apr. 6–7), ACM Press, New York, NY, 33–37.

THORUP, M. 1999. Undirected single-source shortest paths with positive integer weights in linear time. *J. Assoc. Comput. Mach. 46*, 3, 362 – 394.

WINTER, P. 1993. Euclidean Steiner minimal trees with obstacles and Steiner visibility graphs. *Discrete App. Math. 47*, 2, 187–206.

XING, Z. AND KAO, R. 2002. Shortest path search using tiles and piecewise linear cost propagation. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst. 21*, 2 (Feb.), 145–158.

Received June 2003; revised October 2003, December 2003; accepted December 2003