Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, Kunming, 12-15 July 2008

AN IMPROVED METHOD FOR HIDING DATA IN A MAZE

HUI-LUNG LEE, CHIA-FENG LEE, LING-HWEI CHEN

Department of Computer Science, National Chiao Tung University Hsinchu, Taiwan, R.O.C.
E-MAIL: huilung@debut.cis.nctu.edu.tw, encounter@debut.cis.nctu.edu.tw, lhchen@cc.nctu.edu.tw

Abstract:

It is common to use image, audio, video streams as host media in steganography. In this paper, we will select mazes as host media to avoid the problem "robustness" concerned by classic host media. The original idea of embedding data in a maze is proposed by Niwayama et al. Their method has two disadvantages. One is very small embedding capacity; the other is that the embedded maze is not perfect. Here, we present an improved algorithm to increase the embedding capacity. Meanwhile, we can still preserve the property "perfect", that is, we can keep the imperceptible property required in steganography.

Keywords:

Perfect maze; steganography; solution path

1. Introduction

In steganography, images, audios, video streams are often used as host media. In general, these media are usually stored in compression form. Thus if secret data are embedded in such kinds of media, they may be destroyed due to compression. Hence embedding data in such kinds of media must take robustness into account. On the other hand, a maze is defined logically, we do not need to concern the above problem. A maze (see Fig. 1) basically contains cells, walls, a starting cell, and an end cell. Logically, a maze is a complex multipath network and a player is to find a solution path from the starting cell to the end cell. A rectangular maze has m cells in width and n cells in height and is denoted as m × n maze, it is called perfect if there exists one and only one path between any two cells [1]. Figs. 2 and 3 [2] show a perfect maze and an imperfect maze, respectively. Here we only deal with rectangular perfect mazes, since this type is most common [1].

Regarding cells as nodes, carved invisible walls as links, we can express a maze as a graph. Fig. 4 shows an example, a number attached in a link connecting two nodes N and M stands for the number of intermediate nodes appearing in the path from N to M. Based on this expression, we can find that a perfect maze corresponds to a tree (see Fig. 5), such relation can be used to prove whether a generated maze is perfect.

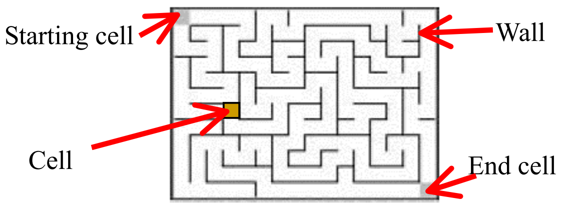

Figure 1. An example to illustrate a maze structure.

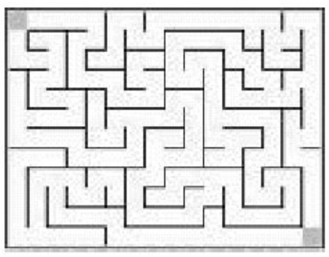

Figure 2. A 16 × 12 perfect maze.

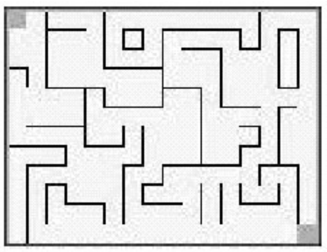

Figure 3. A 16 × 12 imperfect maze.

1.1. Hunt-and-Kill maze generating (HKMG) algorithm

Many algorithms were proposed to generate a maze, Hunt-and-Kill maze generating (HKMG) algorithm [3] is a typical maze generator. Fig. 6 shows a maze generated by the HKMG algorithm. In HKMG algorithm, there are three types of cells defined as follows:

- 'In' cell (I): a cell that has been processed and always keeps its type.
- 'Frontier' cell (F): a cell that is processed and is a four-neighbor of a certain "I" cell.
- 'Out' cell (O): a cell not yet processed.

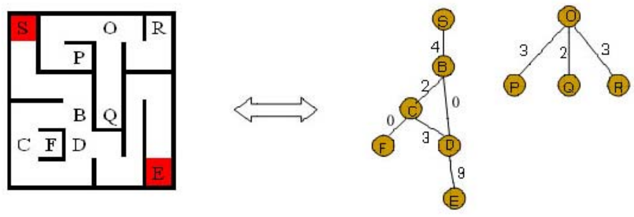

Figure 4. Correspondence between an imperfect maze and a graph.

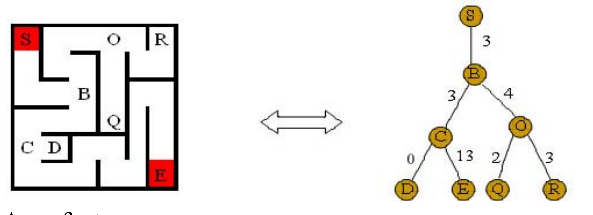

Figure 5. Correspondence between a perfect maze and a tree.

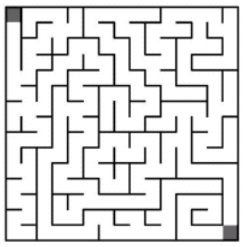

Figure 6. A HK generated

In the following, we will give a brief description for the HKMG algorithm.

(1) Mark all cells as O cells.
(2) Mark the starting cell and mark each O cell in its four-neighborhood as F cell.
(3) Choose an F cell around an I cell and carve the wall between the F cell and the I cell. Mark the F cell as I cell and mark each O cell in its four-neighborhood as F cell. Repeat step 3 until there is no F cell.
(4) End.

As mentioned previously, a maze can be represented by a graph, we can claim that the HKMG algorithm generates a perfect maze by only showing that the corresponding graph is a tree. The reason is that any two nodes in a tree have one and only one path. This matches the property of a perfect maze. In the following, we will give a brief proof. At first, let set A contain only the starting cell. Create an empty graph G and let the starting cell to be the root node of G. In step 3 of the HKMG algorithm, a new node corresponding to the F cell is created and added when an F cell is chosen in G. When a wall between the F cell and its neighbor I cell is carved, a link between the corresponding two nodes (one for I cell in set A, the other for F cell outside A) is added in G, and the F cell is added in set A. This will guarantee that no loop occurs in G. Thus, at the end of the HKMG algorithm, each cell has a corresponding node added to G and connected to a certain node. This means that the corresponding graph G is a tree.

1.2. Hunt-and-Kill embedding algorithm

Niwayama et. al. [3] provided a data hiding method called HK embedding algorithm, which embeds secret data in a maze generated by the HKMG algorithm. The HK embedding algorithm is described as follows:

(1) Locate a solution path from the starting cell to the end cell.
(2) Make branches in the solution path depending on embedded data. Concretely, if the embedded data bit is 1 then make a branch on the right, else if the data bit is 0 then on the left.
(3) Applying HKMG algorithm to complete the maze establishing.

Unfortunately, the HK embedding maze algorithm may generate some inaccessible sections. That is, the generated maze using this algorithm is not a perfect maze. This might attract the attention of inspectors. To overcome these problems, we will propose an improved embedding method to generate a perfect maze and to provide more embedding capacity.

2. The Proposed Method

The main idea of the proposed method is to consider multipaths rather than only the solution path to gain more embedding capacity. Before describing the proposed method,, we will define "embeddable cell" which will be used to embed a bit. Suppose the HKMG algorithm is used to generate a perfect maze, and the solution path from the starting cell to the end cell is located. All cells on the path marked as I cells are the other cells are reset to be O cells, and all walls are rebuilt, except those in the path (see Fig. 7(a)).

Definition 1:

An embeddable cell is an I cell, which is in the solution path with exact two O cells in its four-neighbors and each O cell should not be a neighbor of another I cell in the solution path.

Fig. 7 shows an example. In Fig. 7(b), the cells with black triangles are embeddable ones and the gray cell is the overlapped neighbor of two I cells, thus the cell with a black solid circle is not an embeddable cell. Based on the definition, all embeddable cells can be located. According to the embedding bit, we carve the wall between an embeddable cell and one O cell around it, and mark the O cell as I cell. There are six kinds of embeddable cells shown in Fig. 8. Each embeddable cell has two neighboring O cells marked as 1 and 0. If a "1" bit is embedded, then the wall between the embeddable cell and the cell marked "1" is carved, and the "1" cell is marked as I cell. Otherwise the wall between the embeddable cell and the cell marked as "0" is carved, and the "0" cell is marked as I cell.

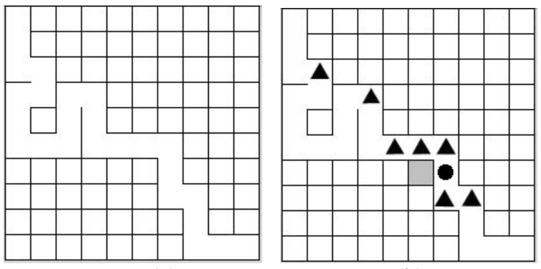

Figure. 7 Two examples to illustrate embeddable cells. (a) The white path stands for the solution path. (b) The cell with a black circle is not an embeddable cell, the cells with black triangles are embeddable cells.

To increase more embedding capacity, we can embed bits into multipaths instead of only one path. In the proposed method, we first generate a perfect maze by HKMG algorithm. Subsequently, we choose some cells as the start cells and one cell as the common end one of the multipaths. Finally, we solve the perfect maze to obtain the corresponding multipaths. These multipaths sometimes will merge at some cells. Fig. 9(a) shows that two paths (one from cell S to cell E, the other from cell T to cell E) merge at M. Fig. 10(a) shows a 30×30 perfect maze generated by the HKMG algorithm. Fig. 10(b) shows four solution paths. The four solution paths start at S, T, K and H cells, respectively. They have the common end cell E. The second path is merged into the first path at cell A, the third path is merged into the first path at cell B, and the fourth path is merged into the second path at cell C.

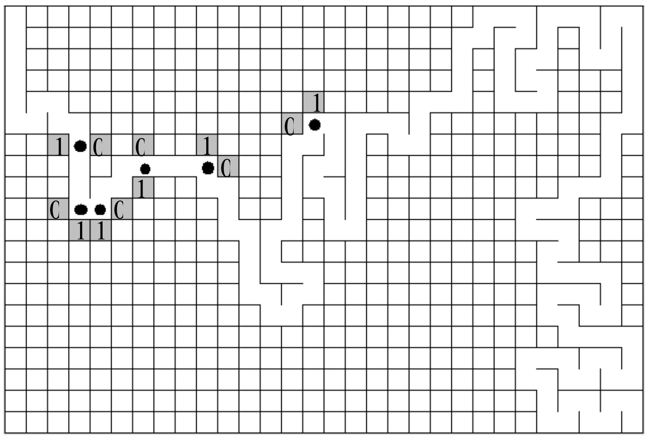

Figure 8. Six kinds of embeddable cells.

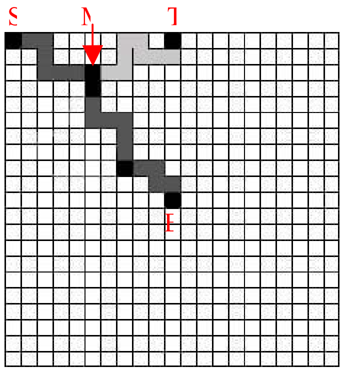

Figure 9. Two paths from cell S and cell T to cell E merging at cell M.

2.1. Embedding algorithm

After obtaining the multiple paths, we order these paths according to their starting cells from top to bottom and then left to right. According to the path sequence, we trace each path from the start cell to locate all embeddable cells. Note that if a subpath of a certain path is ever traced, it will be skipped. As a result, we obtain a sequence of embeddable cells. Then data can be embedded according to this sequence. Here, we will define a new type of cell, 'D' cell, which will be used in the proposed embedding algorithm.

Definition 2:

Let A be an embeddable cell, cells B and C be its two neighboring O cells. If the wall between A and B is carved to embed one bit, cell C is called a D cell.

The details of the proposed embedding algorithm are described as follows:

(1) Create a maze by the HKMG algorithm (see Fig 10(a)).
(2) Choose some cells as start cells and one cell as the end cell. Solve multipaths from these starting cells to the end cell (see Fig. 10(b)).
(3) Reset all cells to be O cells and all walls as visible. Set those cells on multipaths to be I cells. Carve each wall between two I cells (see Fig. 11(a)).
(4) Find out all embeddable cells in each path and order them according to the path sequence. Note that we do not set the boundary to be embeddable cell even if the end cell is embeddable.
(5) For each embeddable cell, if the embedding bit is 1 (0), the wall between the embeddable cell and its neighboring O cell marked 1 (0) (see Fig. 8) is carved and the cell marked 1 (0) is set as I cell, and the other neighboring O cell marked 0 (1) is set as D cell (see Fig. 11(b)).
(6) Set those O cells around I cells to be F cells.
(7) Process these F cells using HKMG algorithm (see Fig. 11(c)).
(8) Process the D cells and those O cells which are surrounded by D cells.
    a、 Scan the maze again. Check if any D cell exists. If none, go to step 9. Otherwise, choose a D cell with one of its neighbors being an unembeddable I cell and carve the wall between the D cell and the unembeddable I cell. Mark the D cell as I cell and mark its four-neighboring O cells as F cells.
    b、 Check if any F cell exists. If none, go to step 8(a).
(9) End.

Fig. 11 shows an example. Fig. 11(a) shows the result after performing the steps 1-3. Fig. 11(b) shows the result after embedding data in embeddable cells, and different colors represent different kinds of embeddable cells. Fig. 11(c) shows the result after performing step 7 on Fig. 11(b). Fig. 11(d) shows the generated maze after processing D cells.

In fact, we have proved that the maze generated by the proposed algorithm is perfect. Due to the page limitation, we omit the proof.

2.2. Extracting algorithm

To extract the embedded bits, the receiver must have obtained the locations of start and end cells of multipaths. This part can be considered as the sharing secret between the sender and receiver. When the receiver obtains the generated maze since the maze is perfect, he can get the original multipaths accurately according to these start and end cells. Then, the receiver can locate all embeddable cells. Finally according to the direction of the branch in each embeddable cell, secret data in bits can be extracted successfully.

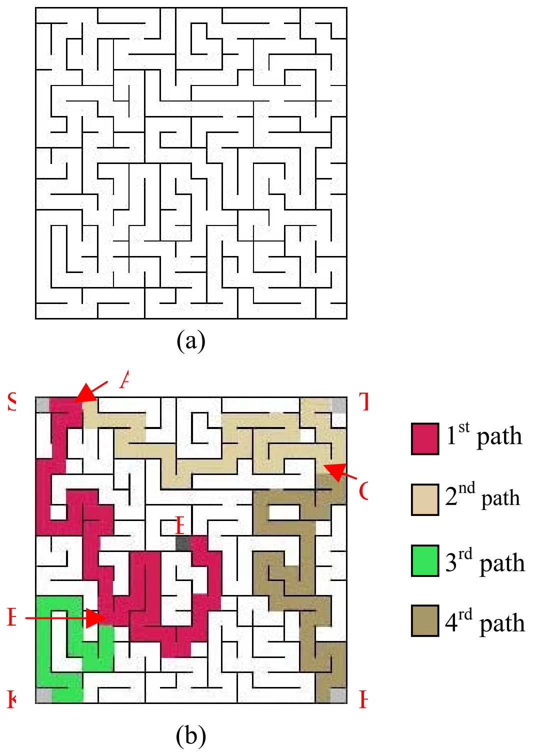

Figure 10. An example of multipaths. (a) A perfect maze generated by HK maze generating algorithm. (b) Four paths in (a) located with merged points A, B, and C.

3. Experimental Results

Here, we will demonstrate two perfect mazes generated by our proposed method. Fig. 12 shows a 30 × 30 maze generated by our proposed method (four paths), in which there are 77 bits embedding capacity. Fig. 13 shows a 50 × 50 maze generated by our proposed (four paths) method in which there is 171 bit embedding capacity. From these two figures, we can see that our proposed algorithm actually provide higher embedding capacity and the generated maze still keeps perfect.

4. Conclusions

Without lost of preserving perfect, our proposed method can provide higher embedding capacity than that of one-path HK embedding maze generating algorithm. As the size of a maze increases, we may try to use more paths to get more capacity while keeping “imperceptibility”.

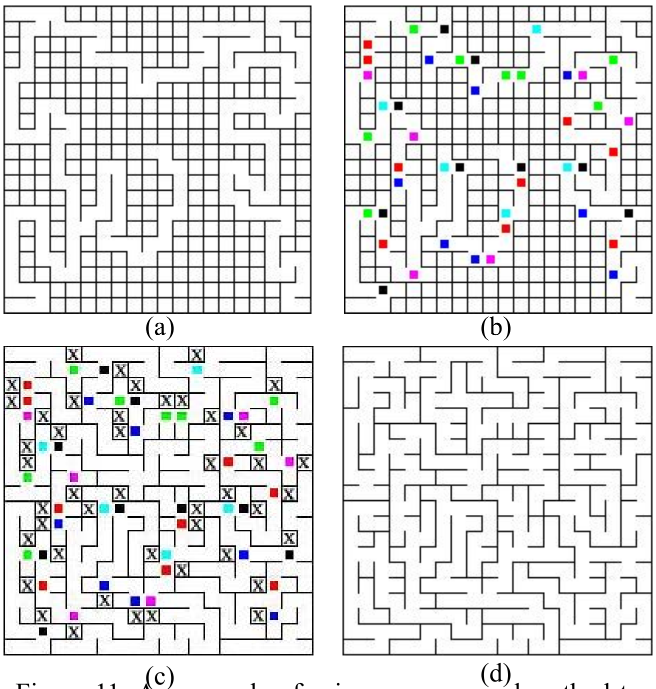

Figure 11. An example of using our proposed method to embed data in a perfect maze. (a) The result after performing Step 3 of the proposed embedding algorithm to Fig. 11(b). (b) The result after embedding data in embeddable cells marked by colors with D cells. (c) The result of applying HKMG algorithm to process F cells and marked D cells by “×”. (d) The perfect maze after processing D cells.

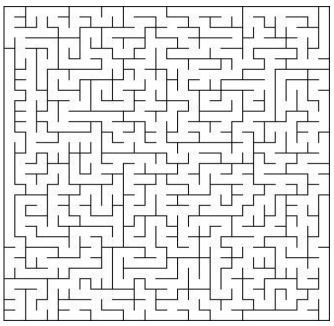

Figure 12. A maze generated by the proposed method with 77 bits embedded.

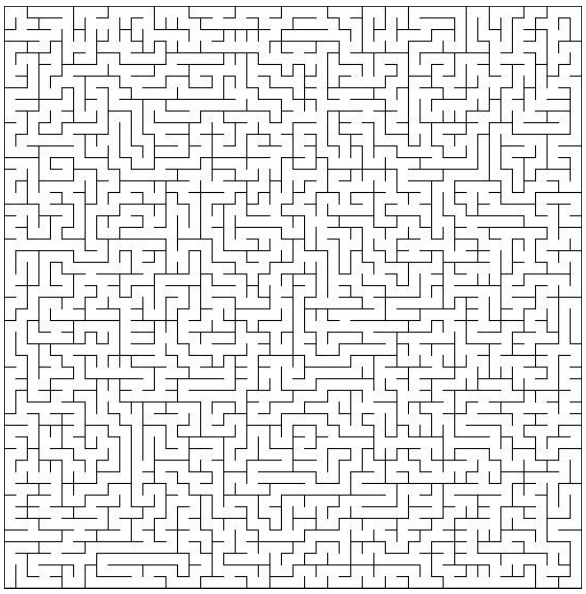

Figure 13. A maze generated by the proposed method with 171 bits embedded.

Acknowledgements

This work is supported in part by National Science Council and Taiwan Information Security Center at NCTU (NSC96-2219-E-009-013-.).

References

[1] W. Pullen, “Think Labyrinth, Maze Algorithms,” http://www.astrolog.org/labyrnth /algrithm.htm.

[2] B. Kirkland, “Maze Works－How to Build a Maze,” http://www.mazeworks.com/mazegen/mazetut.

[3] Naomoto Niwayama, Nasen Chen, Takeshi Ogihara, and Yukio Kaneda, “A Steganographic Method for Mazes, ” Pacific Rim Workshop on Digital Steganography 2002 (STEG'02), Kitakyushu, Japan.