Electronic Preprint for Journal of Information Processing Vol.21 No.3

Regular Paper

A Picturesque Maze Generation Algorithm with Any Given Endpoints

Koki Hamada[1,a)]

Received: July 31, 2012, Accepted: January 11, 2013

Abstract: A picturesque maze is a kind of maze in which the solution path reveals a hidden black-and-white raster image. We propose here an algorithm to generate a picturesque maze of a given black-and-white raster image. Okamoto and Uehara proposed an algorithm to generate a picturesque maze by turning each original pixel into a 2-by-2 set of pixels. One drawback of the method is that the entrance and the exit are always adjacent. We propose a simple algorithm to generate a picturesque maze with any given endpoints for a 2-edge-connected input image.

Keywords: picturesque maze, 2-edge-connectivity, given endpoints

1. Introduction

A picturesque maze is a kind of a maze in which the solution path reveals a hidden black-and-white input image. Okamoto and Uehara formalized the problem as follows [5].

Input A black-and-white raster image with m rows and n columns.

Output A maze in which the solution path fills up the input black pixels.

An example is shown in **Fig. 1**. Okamoto and Uehara also proposed an algorithm to solve the above problem [5]. One of the drawbacks of their method, however, is that the entrance and the exit are always adjacent. This paper proposes a simple algorithm to generate a picturesque maze with any given entrance and exit when the input image is 2-edge-connected.

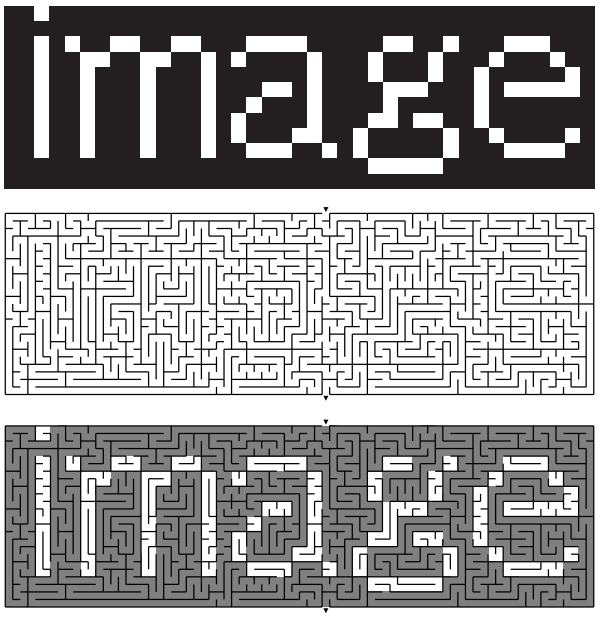

Fig. 1 Example of picturesque maze. (Top) Input black-and-white raster image. (Middle) An output, where the two triangles represent the entrance and the exit. (Bottom) Solution path for the output.

[1] NTT Secure Platform Laboratories, Musashino, Tokyo 180–8585, Japan
[a)] hamada.koki@lab.ntt.co.jp

2. Existing Methods

2.1 General Maze Generation

We begin by introducing a maze generation algorithm that fills a given rectangle with m rows and n columns. The algorithm is as follows.

(1) We construct an undirected graph that corresponds to the given $m \times n$ rectangle. That is, each vertex corresponds to the cell of the rectangle, and an edge $\{u, v\}$ exists when the cells corresponding to u and v are adjacent.
(2) We construct a random spanning tree on the graph.
(3) Then, we remove the sides of cells that correspond to the edges of the spanning tree. By arbitrarily choosing the entrance and the exit cells, we have a random maze.

We note that the solution path is always unique by the properties of a tree.

2.2 Picturesque Maze Generation

Now, we consider the problem to generate picturesque mazes. For a given $m \times n$ black-and-white raster image, we construct a graph that corresponds to the given rectangle as we have done in the above case. This time we have to construct a spanning tree whose solution path is a Hamiltonian path of a subgraph induced by vertices that correspond to black cells (we call this subgraph the *foreground graph*). Unfortunately, some images have no Hamiltonian path even when the foreground graph is connected. In addition, Itai et al. showed that the problem of deciding whether there exists a Hamiltonian path is NP-complete [3]. Umans and Lenhart proposed a polynomial-time algorithm for finding a Hamiltonian path when the image has no holes [7]. However, the condition is too restricted since most of the images used for picturesque mazes have holes.

2.3 Method of Okamoto and Uehara

To overcome the hardness of finding a Hamiltonian path on the foreground graph, Okamoto and Uehara proposed a simple and powerful method to generate picturesque mazes. Their idea is to turn each pixel into a 2-by-2 set of pixels. This yields an image with $2m \times 2n$, and we call this new image a *(2-by-2-)extended image*. As we will see in the following, when this replacement is executed, the foreground graph of the extended image always has at least one Hamiltonian path. In addition, one of the Hamiltonian paths is easily found by applying the following simple algorithm proposed by Okamoto and Uehara [5].

(1) We are given a black-and-white raster image (**Fig. 2** (a)).
(2) We construct a foreground graph of the image (Fig. 2 (b)) and a spanning tree on the foreground graph (Fig. 2 (c)).
(3) We traverse the spanning tree by applying the right-hand rule (Fig. 2 (d)).
(4) Along with the traversal we construct a Hamiltonian path on the extended image (Fig. 2 (e)).
(5) We construct a spanning tree on the foreground graph of the extended image by adding edges to the Hamiltonian path (Fig. 2 (f)). We let the endpoints of the Hamiltonian path be the entrance and the exit of the maze. Now, we have a picturesque maze whose solution path forms the input image.

Thus, this algorithm always finds a Hamiltonian path on the foreground graph of the extended image if the foreground graph of the original input image is connected. One of the properties of this algorithm is that the entrance and the exit of the solution path are always adjacent. This property is useful when the algorithm is used as part of other algorithms (in fact we use this in the following subsection and in Section 3). However, when the entrance and the exit are adjacent, the corresponding maze can be solved without facing any deadends by applying the so-called right-hand rule (**Fig. 3** (a)). This makes the generated mazes too easy, and it is desired that both the right-hand rule and the left-hand rule lead the player to some deadends (Fig. 3 (b) and Fig. 3 (c)).

2.4 Tiling Based Methods

To resolve the restriction of the adjacency of the entrance and the exit, Nakai and Okamoto proposed an algorithm that also takes an arbitrarily chosen entrance and exit as inputs [4]. The algorithm is as follows.

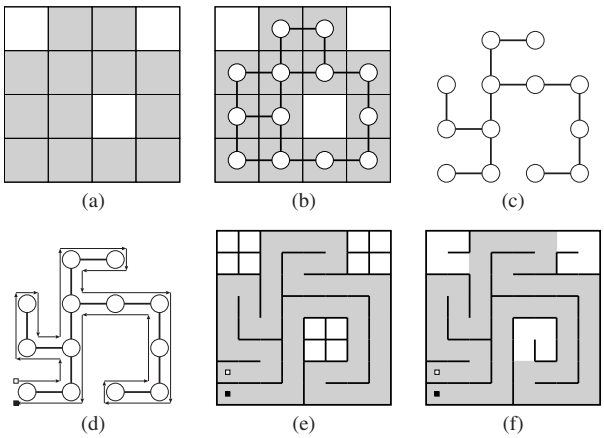

Fig. 2 Execution example of Okamoto and Uehara's method [5].

(1) We are given a black-and-white raster image, the entrance and exit pixels, and the positions of the entrance and the exit (**Fig. 4** (a)). The positions are elements of the set {upper right, upper left, lower right, lower left}, and are used to specify the exact pixel from the corresponding 2-by-2 set of pixels on the extended image. The positions of the entrance and the exit are required to have different parity. That is, one is in the set {upper right, lower left} and the other is in the set {upper left, lower right}.
(2) We construct the foreground graph of the image (Fig. 4 (b)) and find a simple path from the entrance vertex to the exit vertex on the foreground graph (Fig. 4 (c)).
(3) Along with the simple path on the foreground graph we construct a simple path on the 2-by-2-extended image (Fig. 4 (d)). The construction is done by *tiling* a 2-by-2 set of pixels (**Fig. 5**) for each vertex on the path. The tiling is done one by one for every vertex on the path from the entrance to the exit, as shown in **Fig. 6**.
(4) We extend the path on the extended image to a Hamiltonian path of the extended foreground graph by *detouring*. That is,

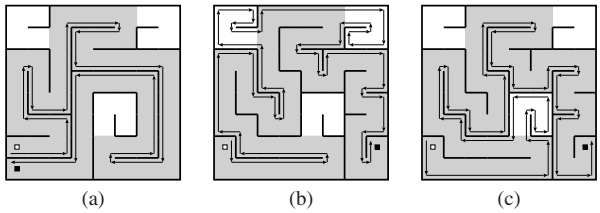

Fig. 3 Comparison of generated mazes.

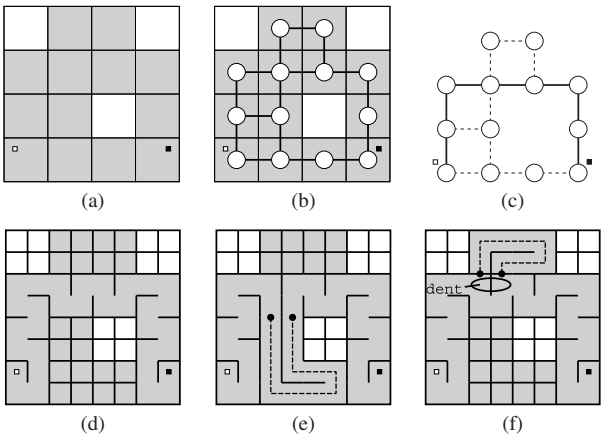

Fig. 4 Execution example of Nakai and Okamoto's method [4].

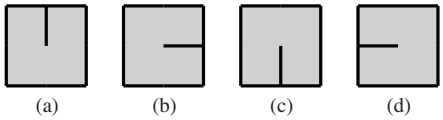

Fig. 5 Tiling patterns for 2-by-2-extended image.

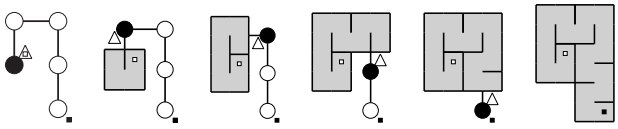

Fig. 6 Step-by-step example of tiling. Small white and black squares represent the entrance and exit vertices and their positions. A black circle and a triangle represent the current vertex and its starting position, respectively.

we apply the method of Okamoto and Uehara [5] to extend the path (Fig. 4 (e)).

The main problem of this algorithm is that the detouring in Step 4 does not always work since detouring is not possible when a *dent* in the solution path is facing another foreground area (Fig. 4 (f)). In addition, some images have no solution path when certain endpoints are given. To avoid this problem, Nakai and Okamoto turns each pixel on the 2-by-2-extended image into a 2-by-2 set of pixels again (i.e., an original pixel is now turned into a 4-by-4 set of pixels).

Ikeda also proposed an algorithm that takes an arbitrarily chosen entrance and exit as input [1]. Ikeda resolved the problem by replacing each pixel with a 3-by-3 set of pixels while allowing some modification of the image.

2.5 Another Method

Ikeda and Hashimoto proposed an algorithm based on another approach [2]. In contrast to the above algorithms [1], [4], [5], the algorithm of Ikeda and Hashimoto does not change the resolution of the image. Instead of changing the resolution, they allow the image to be modified. To minimize the modification, they defined a utility function and minimized it by using simulated annealing.

3. Proposed Method

We propose an algorithm to generate a picturesque maze with given entrance and exit on the 2-by-2-extended image of a given raster image. Our method is also based on the tiling approach, like the method of Nakai and Okamoto [4]. That is, the generation of a solution path consists of the four steps described in Section 2.4. As explained in Section 2.4, the main problem of the tiling approach is that the detouring in Step 4 does not always succeed. To resolve this problem, we based our method on the following two ideas.

- To ensure the existence of Hamiltonian paths for any pair of endpoints, we restrict the input image to be *2-edge-connected*.
- In Step 2, we carefully choose a path from s to t on the foreground graph so that we can always extend the path by detouring in Step 4.

3.1 2-edge-connectivity

We assume that the foreground graph of the input raster image is 2-edge-connected. A graph is 2-edge-connected if it is connected and contains at least two vertices but no bridge. A bridge of a connected graph is an edge such that if the edge is removed from the graph, the graph becomes disconnected. Tarjan proposed an algorithm that finds all bridges in $O(n)$ time where n denotes the number of vertices [6]. Thus, we can efficiently test whether the foreground graph of a given image is 2-edge-connected or not.

3.2 Details of Our Method

As previously mentioned, our method is based on the tiling method. That is, we accept an image and endpoints as inputs (**Fig. 7** (a)), construct the foreground graph of the image (Fig. 7 (b)), find a path on the foreground graph by constructing a depth-first-search tree (Fig. 7 (c)), construct a path on the 2-by-2-extended image by tiling (Fig. 7 (d)), extend the path to a Hamiltonian path by detouring (Fig. 7 (e)), and then, extend the Hamiltonian path to a spanning tree by adding edges (Fig. 7 (f)). The difference is the step of finding a simple path from the entrance vertex to the exit vertex on the foreground graph (Fig. 7 (c)). Therefore, we only describe here how to find that simple path.

We begin by defining the input and output of our method as follows.

Input A foreground graph $G = (V_G, E_G)$ of a black-and-white raster image, vertices $s, t \in V_G$, which represent the entrance and the exit, and the positions of s, t where G is 2-edge-connected and the positions have different parity.

Output A Hamiltonian path on the foreground graph of the 2-by-2-extended image where the endpoints are pixels on the extended image and are specified by s, t, and their positions.

Now, we describe our algorithm to find a path on the foreground graph. Intuitively, to find a path on the foreground graph, we conduct a depth-first search to find a path from s to t resulting in as few dents as possible. The details of our method are as follows.

(1) We are given a foreground graph $G = (V_G, E_G)$ of the input raster image, the entrance and exit vertices $s, t \in V_G$, and their positions p_s, p_t.
(2) We color all vertices V_G WHITE.
(3) We start a depth-first search by visiting the entrance vertex s that has position p_s. When a vertex $u \in V_G$ that has position p_u is visited, the following procedure is recursively conducted.
  (a) We color the vertex u GRAY.
  (b) We recursively visit every adjacent unvisited vertex of u in order. The order is determined from the position p_u of u according to **Table 1**. For example, if the position p_u is at the upper right, we first try to visit the adjacent vertex on the right, then the upper, left, and lower vertices. An illustration is shown in **Fig. 8**.
  (c) If $u = t$, we leave this depth-first search.
  (d) We color the vertex u BLACK.
(4) We output the path P from s to t on the depth-first-search tree.

Since the algorithm is a kind of depth-first search, a *depth-first-*

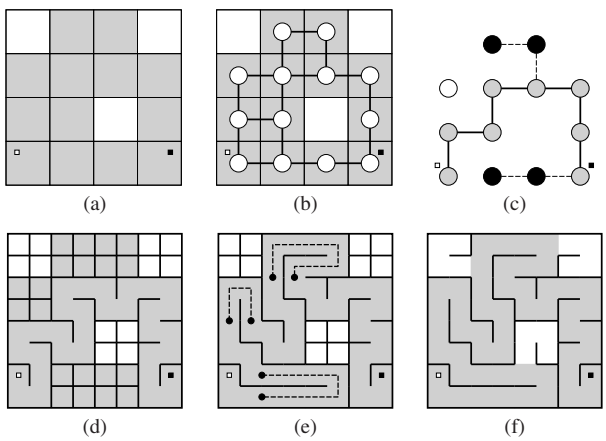

Fig. 7 Execution example of our method.

Table 1 Ordered list of directions of the next vertex to be visited when the current vertex's position is upper right. If the position is not upper right, each element is rotated. The abbreviations "LL" and "UR" in the table represent "lower left" and "upper right," respectively.

Order	1	2	3	4
Direction of the next vertex	Right	Upper	Left	Lower
Tiling pattern	Fig. 5 (b)	Fig. 5 (a)	Fig. 5 (a)	Fig. 5 (b)
Position of the next vertex	LL	LL	UR	UR
Direction of the dent	Right	Upper	Upper	Right

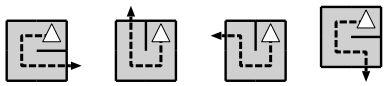

Fig. 8 Illustration of the ordered list of directions when the current vertex's position is upper right. The small triangle represents the current vertex's position, the arrow represents the direction of the next vertex, and the corresponding tiling pattern is shown.

search tree is constructed during the execution of the algorithm. Initially, the depth-first-search tree contains its root vertex s only. Whenever an unvisited vertex v is visited from another vertex u, the vertex v and the edge $\{v, u\}$ are added to the tree. We say that u is the *predecessor* of v in the depth-first-search tree and describe the predecessor of v as $\pi(v)$.

We note that the running time of the algorithm is linear in the number of pixels in the image since every vertex in V_G is visited at most once.

3.3 Correctness of Our Method

Now, we prove the correctness of our method.

Theorem 1. *When a foreground graph $G = (V_G, E_G)$ and its vertices s, t are given, the proposed algorithm outputs a simple path P from s to t on G.*

Proof. When a vertex $u \in V_G$ is visited, the algorithm recursively visits the adjacent vertices of u and never returns until the vertex t has been visited or all the adjacent vertices of u have been visited. Since the foreground graph G is connected, this recursive search continues until the vertex t has been visited. Since the depth-first-search tree is a tree and includes s and t, the output is a simple path from s to t. □

Next, we prove that the detouring always succeeds when a simple path P from a vertex s to a vertex t on the foreground graph G is computed. Let V_P be the set of vertices included in path P. As Fig. 6 shows, when path P and the positions of vertices s, t are given, the tiling is uniquely determined.

The detouring is conducted for each connected component C on the subgraph of G induced by the vertices $V_G \setminus V_P$. As Fig. 4 implies, the detouring succeeds when there exists an edge $\{g, v\} \in E_G$ such that $g \in V_P$, $v \in V_C$ and $\mathrm{Dent}(g) \neq v$. We write $\mathrm{Dent}(g) = x$ if x is a vertex adjacent to g and is facing the dent when g is tiled. If such a vertex does not exist, $\mathrm{Dent}(g) = \textsc{nil}$. We prove that the above condition holds for every connected component, that is, our method always outputs a Hamiltonian path on the extended image.

Theorem 2. *Let $C = (V_C, E_C)$ be a connected component on the subgraph of G induced by the vertices $V_G \setminus V_P$. Then, $\exists\{g, v\} \in E_G$ s.t. $g \in V_P$, $v \in V_C$ and $\mathrm{Dent}(g) \neq v$.*

Proof of Theorem 2. We use $\mathrm{Color}(x)$ to denote the color of the vertex $x \in V_G$. From the property of the depth-first-search tree, we can easily confirm that $\mathrm{Color}(u) = \textsc{gray}$ if and only if $u \in V_P$.

When the color of a vertex $u \in V_G$ is set to $\textsc{black}$, $\mathrm{Color}(v) \neq \textsc{white}$ for every vertex v adjacent to u.

Since C is connected and $V_C \cap V_P = \emptyset$, $\mathrm{Color}(u) = \mathrm{Color}(v) \neq \textsc{gray}$ for every $u, v \in V_C$. That is, one of the following conditions $\forall c \in V_C$, $\mathrm{Color}(c) = \textsc{white}$ or $\forall c \in V_C$, $\mathrm{Color}(c) = \textsc{black}$ holds. Let $\mathcal{B}$ be the set of connected component $B = (V_B, E_B)$ on the subgraph of G s.t. $\forall c \in V_B$, $\mathrm{Color}(c) = \textsc{black}$, and $\mathcal{W}$ be the set of connected component $W = (V_W, E_W)$ on the subgraph of G s.t. $\forall c \in V_W$, $\mathrm{Color}(c) = \textsc{white}$. Then, $C \in \mathcal{B}$ or $C \in \mathcal{W}$ holds. Now we consider the following Lemma 1 and Lemma 2.

Lemma 1. *$\forall W = (V_W, E_W) \in \mathcal{W}$, $\exists\{w, g\}$ s.t. $w \in V_W$, $g \in V_P$, $\{w, g\} \in E_G$ and $\mathrm{Dent}(g) \neq w$.*

Proof of Lemma 1. Since the foreground graph G is connected, there exists an edge $\{w, g\} \in E_G$ s.t. $w \in V_W$ and $g \in V_P$. By the order of adjacent vertices to be visited in Table 1, $\mathrm{Dent}(g) = \textsc{nil}$ or $\mathrm{Color}(\mathrm{Dent}(g)) \neq \textsc{white}$ holds. Therefore, $\mathrm{Color}(w) = \textsc{white}$ implies that $\mathrm{Dent}(g) \neq w$. □

Lemma 2. *$\forall B = (V_B, E_B) \in \mathcal{B}$, $\exists\{b, g\}$ s.t. $b \in V_B$, $g \in V_P$, $\{b, g\} \in E_G$ and $\mathrm{Dent}(g) \neq b$.*

Proof of Lemma 2. We begin by considering the following lemma.

Lemma 3. *Let $B = (V_B, E_B) \in \mathcal{B}$. The subgraph of the depth-first-search tree induced by V_B is a tree and there exists exactly one vertex $b_r \in V_B$ such that $\mathrm{Color}(\pi(b_r)) = \textsc{gray}$.*

Proof of Lemma 3. Let B_π be the subgraph of the depth-first-search tree induced by V_B. Suppose that B_π consists of at least two connected components $B_1, B_2, \ldots, B_\ell$ ($\ell \geq 2$). Then, each connected component B_i is a tree since B_i is a subgraph of a tree. Since B is connected, there exists a pair of vertices (b_i, b_j) such that $b_i \in V_{B_i}$, $b_j \in V_{B_j}$, $\{b_i, b_j\} \in E_B$ for two connected components $B_i = (V_{B_i}, E_{B_i})$ and $B_j = (V_{B_j}, E_{B_j})$. Without loss of generality, we assume that $\mathrm{Time}(b_i) < \mathrm{Time}(b_j)$ where $\mathrm{Time}(x)$ represents the time when the vertex $x \in V_G$ is visited. At the time $\mathrm{Time}(b_i)$, $\mathrm{Color}(b_j) = \textsc{white}$ and this implies $b_j \in V_{B_i}$. This contradicts $V_{B_i} \cap V_{B_j} = \emptyset$. Therefore, B_π consists of exactly one tree. Since B_π is part of the depth-first-search tree, there exists exactly one vertex $b_r \in V_B$ that satisfies $\mathrm{Color}(\pi(b_r)) = \textsc{gray}$. □

Since Lemma 3 holds, let b_r be a vertex in V_B s.t. $\mathrm{Color}(\pi(b_r)) = \textsc{gray}$. Then, $\pi(b_r) \in V_P$ holds. Let r be $\pi(b_r)$. Since G is 2-edge-connected, there exists at least one edge between the vertices V_B and $V_G \setminus V_B$ other than $\{b_r, r\}$. We call this edge $\{b, g\}$ where $b \in V_B$ and $g \in V_G \setminus V_B$.

(i) In the case $r = g$, $\{b_r, r\} \neq \{b, g\}$ implies $b_r \neq b$. Since $r = g$, we have that $\mathrm{Dent}(r) = \mathrm{Dent}(g)$. Therefore, at least one of $\mathrm{Dent}(r) \neq b_r$ or $\mathrm{Dent}(g) \neq b$ holds.

(ii) In the case $r \neq g$, since $g \notin V_B$, $\mathrm{Color}(g) \neq \textsc{white}$ at the time $\mathrm{Time}(b_r)$. In addition, $\mathrm{Color}(g) = \textsc{gray}$ now. Therefore $\mathrm{Color}(g) = \textsc{gray}$ also holds at the time $\mathrm{Time}(b_r)$. On the other

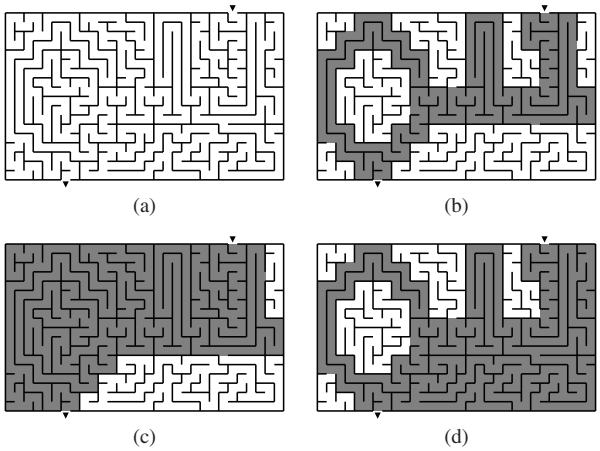

Fig. 9 Example of maze generated by our method.

hand, every $v \in V_G$ s.t. Color(v) = GRAY at the time Time(b_r) satisfies Time(v) ≤ Time(r). Since $r \neq g$, Time(g) < Time(r). At the time Time(r), Color(b) = WHITE, and this implies Dent(g) ≠ b. Dent(g) ≠ b holds until the execution of the algorithm is finished.

Thus, the statement holds in both cases. □

Since Lemma 1 and Lemma 2 hold, the statement of the theorem also holds. □

3.4 Example of Generated Maze

To see the property of mazes generated by our method, we show a maze generated by our method (**Fig. 9** (a)) and its solution (Fig. 9 (b)). The cells visited by applying the right-hand rule and the left-hand rule are shown in Fig. 9 (c) and Fig. 9 (d), respectively. Thus, depending on the input image and endpoints, our method may generate a maze such that both the right-hand rule and the left-hand rule lead the player to some deadends.

4. Conclusion

We proposed a simple algorithm to generate a picturesque maze with any given endpoints for a given 2-edge-connected black-and-white raster image. The proposed algorithm only changes the resolution of the image by turning each pixel into a 2-by-2 set of pixels, as in the method proposed by Okamoto and Uehara [5]. The running time of the proposed algorithm is linear in the number of pixels of the input image.

Acknowledgments We would like to thank the reviewers for their careful and detailed comments which have greatly improved the presentation of this paper.

References

[1] Ikeda, K.: Kaigateki meiro seisei no aru kakuchou (in Japanese) (2010), available from ⟨http://www.lab2.kuis.kyoto-u.ac.jp/~itohiro/Games/100301/100301-10.ppt⟩.
[2] Ikeda, K. and Hashimoto, J.: Stochastic Optimization for Picturesque Maze Generation (in Japanese), *IPSJ Journal*, Vol.53, No.6, pp.1625–1634 (2012).
[3] Itai, A., Papadimitriou, C.H. and Szwarcfiter, J.L.: Hamilton Paths in Grid Graphs, *SIAM J. Comput.*, Vol.11, No.4, pp.676–686 (1982).
[4] Nakai, R. and Okamoto, Y.: Kaigateki meiro sakusei arugorizumu no kaizen (in Japanese), *Mathematical Foundation of Algorithms and Computer Science*, Zhou, X. (Ed.), RIMS Kokyuroku, Vol.1691, pp.162–166, Kyoto University (2010).
[5] Okamoto, Y. and Uehara, R.: How to make a picturesque maze, *CCCG*, pp.137–140 (2009).
[6] Tarjan, R.E.: A Note on Finding the Bridges of a Graph, *Inf. Process. Lett.*, Vol.2, No.6, pp.160–161 (1974).
[7] Umans, C. and Lenhart, W.: Hamiltonian Cycles in Solid Grid Graphs, *FOCS*, pp.496–505, IEEE Computer Society (1997).

Koki Hamada was born in 1984. He received his B.E. and M.I. degrees from Kyoto University, Kyoto, Japan, in 2007 and 2009, respectively. In 2009, he joined Nippon Telegraph and Telephone Corporation. Currently, he is a researcher of NTT Secure Platform Laboratories. He is presently engaged in research on cryptography and information security.