Solving Route Planning Using Euler Path Transform

Yi-Chong Zeng
Institute of Information Science, Academia Sinica, Taiwan
yichongzeng@iis.sinica.edu.tw

***Abstract*— This paper presents a method to solve route planning problem in maze game and waste collection using Euler path transform, the core technique is mixed-integer linear programming (MILP). There are three issues are considered in the transformation, including, the direction of edge, the limited number of edge, and the user-defined starting and ending vertices. The experiment results show that the proposed method is capable of finding the optimal result for route planning.**

***Index Terms*—Euler path, route planning, mixed-integer linear programming**

I. Introduction

In 1736, the great mathematician Leonhard Euler published a paper to solve the problem of seven bridges of Königsberg, and he translated it into the graph theory problem [1]. Recently, research workers study Euler path problem, it applies on various fields, i.e., DNA sequence alignment, minimal de Bruijn sequence construction, route planning, image processing, etc.

For Eulerian directed graph with an arc-labeling, Matamala and Moreno proposed the algorithm to construct the minimal Eulerian trail and minimal de Bruijn sequence [2]. Zhang and Waterman used Euler path approach in order to avoid the expensive computation of multiple sequence alignment for a large number of DNA sequence [3]. The computational time and the amount of memory size for Zhang and Waterman's approach become approximately linear to size of sequence analyzed. Qian and Yasuhara studied the drawing order recovering of handwritten image to find the smoothest path covered all edges of graph [4]. The direction context was exploited to calculate the smoothness between the edges, and the smoothest path can be found by solving the optimal Euler path problem. Furthermore, Euler path transform can solve the route planning, and the route planning has various calls, such as, the sweeping problem [5], the waste collection planning [6], the periodic vehicle routing problem [7], etc. The objective of route planning is to find the proper route in a graph.

In [6], Shih and Lin proposed a multiple criteria optimization approach, which employs integer programming to solve the problem of route planning for waste collection. Integer programming is one kind of linear programming, which is defined as the problem of maximizing or minimizing a linear function subjected to linear constraints [8]. In this paper, the proposed method utilizes mixed-integer linear programming (MILP) to estimate the number of passed edge in order to implement Euler path transform. Then, the proposed method is applied to two real world cases: waste collection and maze game.

The rest of this paper is organized as follows: Section II describes the definitions of Euler circuit and Euler path. Sections III and VI introduce the problem description and the proposed method, respectively. The experiment results are shown in Section V, and the conclusions are made in Section VI.

II. Euler Circuit and Euler Path

Euler's well-known theorem is described: 'if a graph has an Euler circuit, then every vertex of the graph has even degree' [1]. This theorem is further exactly described: 'if every vertex of a nonempty graph has even degree and the graph is connected, then the graph has an Euler circuit'. Hence, the graph of Euler circuit must satisfy two conditions: (1) the graph is nonempty, and (2) all vertices of the graph have the even degrees. The nonempty graph means that a simple path exists to connect any two distinct vertices of the graph. For the second condition, the degree means the number of edge connected to a vertex. On the other hand, every vertex of the graph is connected with even number of edges.

The Euler path is a sequence of adjacent edges from two different vertices (for example, vertex v_k and vertex v_l, and $k \neq l$), which the sequence starts at v_k and ends at v_l. The corollary is described: 'there is an Euler path from v_k to v_l if, and only if, the graph is connected and those two vertices have odd degrees, and the rest of the vertices have even degrees' [1]. Hence, the graph of Euler path must satisfy three conditions: (1) the graph is nonempty, (2) two vertices have odd degrees, and (3) the other vertices have even degrees. Therefore, the difference between Euler circuit and Euler path is that two vertices have odd degrees. Removing one edge in a graph of Euler circuit, the graph will become an Euler path.

III. Problem Description

The objective of this paper is to find the optimal route by Euler path transform. The proposed method estimates the number of every edge which will be passed through, and then our method is applied to two real world cases: waste collection and maze game. For waste collection, the vehicle drives through the selected streets under the constraint of lowest oil consumption which associates with the sum of edge length. One-way street and two-way street correspond to directional edge and non-directional edge, respectively, therefore, edge direction must be considered in case. For maze game, the entrance and the exit correspond to the starting vertex and the ending vertex, respectively, and those two vertices are set initially. After implementing Euler path transform on maze game, the resultant of maze game is the linkage of the edges which are only passed one time.

IV. Proposed Method

The nonempty graph is composed of vertices and edges, and we define two types of edges, namely non-directional edge and directional edge. Furthermore, the directional edges are classified into two terms depended on vertex, namely the imported edge and the exported edge. An example is shown in Fig.1(a), the directional edge e_4 connects two vertices v_2 and v_3. The edge e_4 is exported at v_3, thus, it is called the exported edge at v_3. On the contrary, the edge e_4 is imported at v_2, thus, it is called the imported edge at v_2. In this study, we emphasize on the general form that the graph is composed of both non-directional and directional edges. An example is shown in Fig.1(a). Euler path transform is conducted via three phases, including, pseudo edge setting, number estimation of passed edge, and edge direction determination. The detail of every phase is described below.

A. Pseudo Edge Setting

As aforementioned in Section II, the difference between Euler circuit and Euler path is that two vertices of a graph have odd degrees and the others vertices have even degrees in Euler path. Hence, these two vertices are connected with odd numbers of edges. All edges in the graph of Euler circuit have even degrees. Once we remove one edge in the graph of Euler circuit, the graph will become Euler path because two vertices between the removed edge have odd degrees. Under this circumstance, we initially set a pseudo edge between the two specified vertices as shown in Fig.1(a) and the pseudo edge is only passed one time. After estimating the number of edge which is passed through, the graph will become Euler circuit. Subsequently, the pseudo edge is removed and then the Euler path is obtained.

It is worth to notice that the direction of pseudo edge determines the starting and ending points in Euler path. For example, Fig.1(a) shows that two vertices v_1 and v_3 are set as the starting vertex and the ending vertex, respectively. Then, we must assign the direction of the pseudo edge, e_{ps}, which starts from v_3 to v_1.

B. Number Estimation of Passed Edge

Assume that a graph is composed of M edges and N vertices. The variables α_i and β_i in (1) denote the numbers of the i-th non-directional edge and the i-th directional edge which will be passed, respectively. The variable A_j represents the total number of the non-directional edge at vertex v_j, and B_j is the difference of the total number of exported edge subtracts that of imported edge at vertex v_j. For instance, the first vertex v_1 has A_1=3 and B_1=−1 as shown in Fig.1(a). The equations are defined as follows:

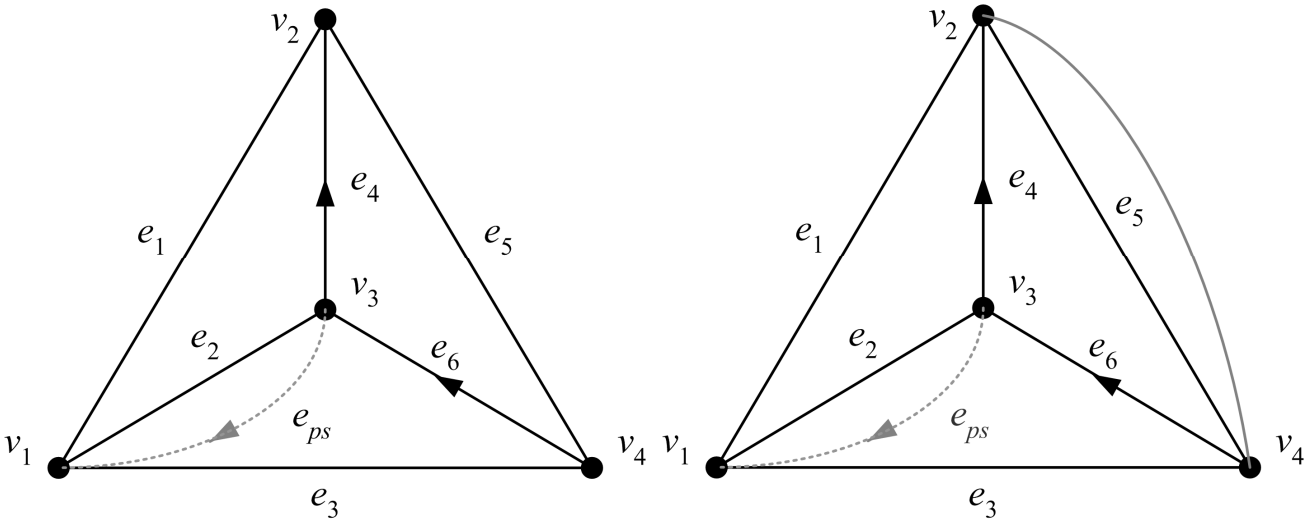


Fig.1. Number Estimation of Passed Edge: Pseudo edge e_{ps} connects two vertices v_1 and v_3, and Euler path will start at v_1 and end at v_3. (a) Original graph, and (b) the gray line between v_2 and v_4 represents e_5 will be passed through once again.

$$\begin{aligned}
&\text{minimize } \sum_{i=0}^{M-1} c_i \cdot (\alpha_i + \beta_i) \\
&\text{subject to} \left(\sum_{i=0}^{M-1} p_{i,j} \cdot \alpha_i + A_j \right) + \left(\sum_{i=0}^{M-1} q_{i,j} \cdot \beta_i + B_j \right) = 2d_j, \\
&\qquad \alpha_i \in \{0,\ 1\}, \\
&\qquad \beta_i \geq 0, \\
&\qquad p_{i,j} \in \{0,\ 1\}, \\
&\qquad q_{i,j} \in \{-1, 0,\ 1\}, \\
&\text{and } j = \{0, 1, ..., N-1\}.
\end{aligned} \tag{1}$$

where c_i is the length of the i-th edge. $p_{i,j}$ represents the relationship between the non-directional edge e_i and vertex v_j. Similarly, $q_{i,j}$ represents the relationship between the directional edge e_i and vertex v_j. d_j is a non-negative integer to ensure that the vertex v_j has even degree. If the i-th non-directional edge connects the j-th vertex, the variable is set to $p_{i,j}$=1; otherwise, $p_{i,j}$=0. If the i-th directional edge connects the j-th vertex, the variable is set to either $q_{i,j}$=–1 for exported edge or $q_{i,j}$=1 for imported edge; otherwise, $q_{i,j}$=0. Once i-th edge is the pseudo edge, both of $p_{i,j}$ and $q_{i,j}$ are set to zero. To analyze (1), it is the form of mixed-integer linear programming; hence, the above equations can be simplified,

$$\begin{aligned}
&\text{minimize } \mathbf{c}^{\mathrm{T}}\mathbf{x} \\
&\text{subject to } \mathbf{A}_{\mathrm{eq}}\mathbf{x} = \mathbf{b}_{\mathrm{eq}}, \\
&\qquad \mathbf{x}_{\mathrm{L}} \leq \mathbf{x} \leq \mathbf{x}_{\mathrm{U}}
\end{aligned} \tag{2}$$

where $\mathbf{x}_{\mathrm{L}}$ and $\mathbf{x}_{\mathrm{U}}$ represent the lower-bound column vector and upper-bound column vector of $\mathbf{x}$, respectively. For the example of Fig.1(a), the graph is composed of 4 vertices and 7 edges which are 4 non-directional edges, 2 directional edges and 1 pseudo edge, and the parameters are set as follows:

$$\begin{aligned}
\mathbf{x} &= [\alpha_1, \alpha_2, \ldots, \alpha_7, \beta_1, \beta_2, \ldots, \beta_7, d_1, d_2, \ldots, d_4]^{\mathrm{T}}, \\
\mathbf{A}_{\mathrm{eq}} &= \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & -2 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -2
\end{bmatrix}, \\
\mathbf{b}_{\mathrm{eq}} &= [-2 \quad -1 \quad -2 \quad -3]^{\mathrm{T}}, \\
\mathbf{c} &= [6 \quad 5 \quad 6 \quad 5 \quad 6 \quad 5 \quad 0 \quad 6 \quad 5 \quad 6 \quad 5 \quad 6 \quad 5 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^{\mathrm{T}}, \\
\mathbf{x}_{\mathrm{L}} &= [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -\infty \quad -\infty \quad -\infty \quad -\infty]^{\mathrm{T}}, \\
\mathbf{x}_{\mathrm{U}} &= [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad \infty \quad \infty \quad \infty \quad \infty \quad \infty \quad \infty \quad \infty \quad \infty \quad \infty \quad \infty]^{\mathrm{T}}
\end{aligned} \tag{3}$$

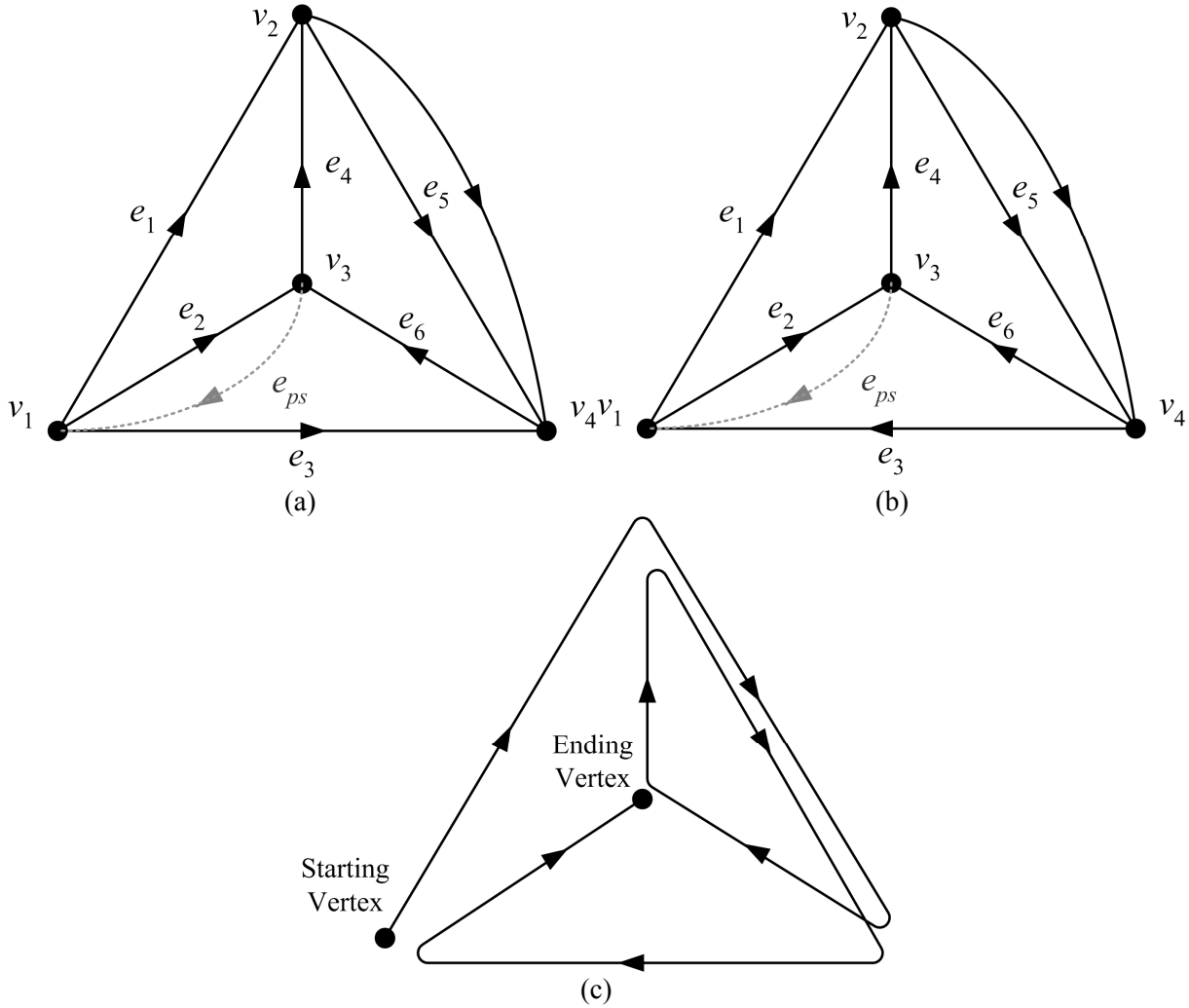


Fig.2. Edge Direction Determination: (a) Initial directions of the graph, (b) the correct directions of the resultant graph, and (c) one of the possible routes.

Mixed-integer linear programming is utilized to solve (3), we estimate the solution of **x**, which is $\mathbf{x}=[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2]^{\mathrm{T}}$. For $\alpha_5=1$, it means that only the non-directional edge e_5 needs to be passed once again, which is represented as a gray line in Fig.1(b). Hence, e_5 is passed twice totally.

C. Edge Direction Determination

The beginning of edge direction determination is to initialize the direction to all non-directional edges. To assign the initial direction, we assume that the edge e_i connects two vertices v_k and v_l, and the direction of the edge starts from v_k and v_l, where $k < l$. The initial direction of Fig.1(b) is illustrated in Fig.2(a). Let $r_{i,j}$ represent the direction of the i-th edge at the j-th vertex, and ϕ_i be the modification coefficient of the i-th edge. If $r_{i,j}=1$, it implies the i-th edge is imported at the j-th vertex. On the contrary, if $r_{i,j}=-1$, it implies the i-th edge is exported at the j-th vertex. In addition, $\phi_i=0$ means that the direction of i-th edge needs to be reversed; otherwise, we maintain the direction of the i-th edge as $\phi_i=1$. Using mixed-integer linear programming, ϕ is solved and the edge direction is determined. The equations are defined as follows:

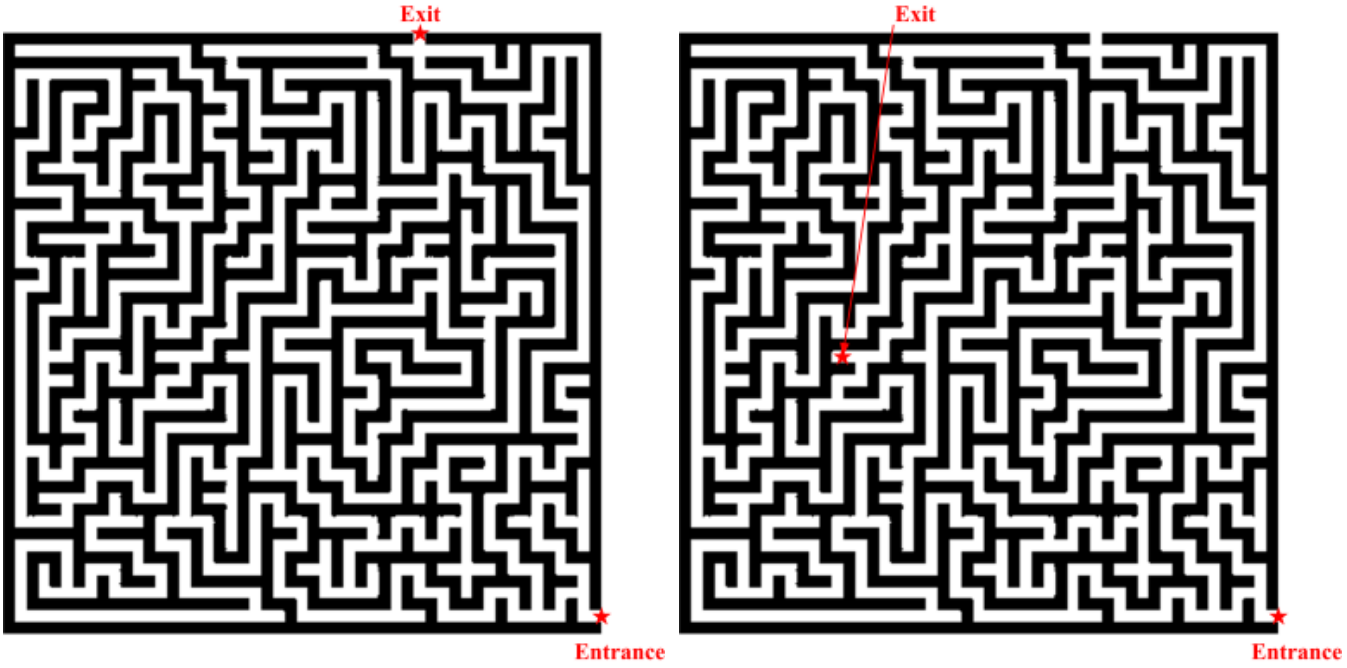
$$\begin{aligned} &\text{maximum} \sum_{i=0}^{M'-1} \phi_i \\ &\text{subject to} \sum_{i=0}^{M'-1} r_{i,j} \cdot (2\phi_i - 1) + B'_j = 0, \\ &\qquad \phi_i \in \{0,1\}, \\ &\text{and } j = \{0,1,\ldots,N-1\} \end{aligned} \tag{4}$$

Fig.3. Route Planning for Waste Collection: (a) Red lines represent the selected streets where the vehicle drives; (b) initial graph; and (c) the final route is found by using the proposed method.

where M' denotes the total number edge of the resultant graph, B'_j is the difference of the total number of exported edge subtracts that of imported edge at vertex v_j. Consequently, Fig.2(a) only needs to reverse the direction of e_3, and the resultant is shown in Fig.2(b). In Fig.2(c), it illustrates one of possible route.

V. Experiment Results

The first experiment is to find the optimal route for vehicle in waste collection. Fig.3(a) shows that the selected streets illustrated by red lines. The graph is composed of 21 vertices and 30 non-directional edges as displayed in Fig.3(b), and this graph is transformed to Euler path. The path



(a) (b)

Fig.4. Maze Game [9]: (a) Maze 1 and (b) Maze 2 have the same entrances but the exits are different to each other.

starts at the top-left corner (v_1) and ends at the bottom-left corner (v_6). The final route planning is shown in Fig.3(c) and the computing time is 2.08 sec. Ten edges pass twice, including, e_3, e_5, e_6, e_{14}, e_{15}, e_{17}, e_{20}, e_{24}, e_{27} and e_{29}. The path is listed below,

$$v_1(e_1)v_2(e_3)v_7(e_{10})v_9(e_{14})v_{12}(e_{18})v_{15}(e_{24})v_{16}(e_{23})$$
$$v_{14}(e_{17})v_{11}(e_{11})v_7(e_3)v_2(e_2)v_3(e_5)v_8(e_{12})v_9(e_{14})v_{12}(e_{19})$$
$$v_{13}(e_{20})v_{18}(e_{25})v_{16}(e_{24})v_{15}(e_{16})v_{11}(e_{17})v_{14}(e_{22})v_{19}(e_{29})$$
$$v_{20}(e_{26})v_{17}(e_{15})v_{10}(e_7)v_4(e_6)v_5(e_9)v_{21}(e_{27})v_{17}(e_{21})$$
$$v_{13}(e_{20})v_{18}(e_{28})v_{19}(e_{29})v_{20}(e_{30})v_{21}(e_{27})v_{17}(e_{15})v_{10}(e_{13})$$
$$v_8(e_5)v_3(e_4)v_4(e_6)v_5(e_8)v_6. \quad (5)$$

In second experiment, two mazes are tested. Those two mazes shown in Figs.4(a) and (b) have the same entrances, but the exits are different to each other. The graph of the maze is composed of 142 edges and 143 vertices. The Euler path and final route of Fig.4(a) are shown in Figs.5(a) and (b), respectively. The process time is 126.01 second. Similarly, the Euler path and final route of Fig.4(b) are shown in Figs.6(a) and (b), respectively. The process time is 63.23 second. For both Fig.5(a) and Fig.6(a), the green line represents the edge has been passed twice or more times; the red line is the final desired resultant. Furthermore, the resultants of four test mazes and are shown in Fig.7.

VI. Conclusions

We propose a method to transform graph to Euler path by using mixed-integer linear programming. The user can self-define the starting vertex and the ending vertex in the graph. The proposed method solves the problem of route planning in order to two applications: the maze game and the waste collection. The experiment results demonstrate the effectiveness of our method. All programs are executed in MATLAB software using a 1.5GHz Pentium-M processor.

References

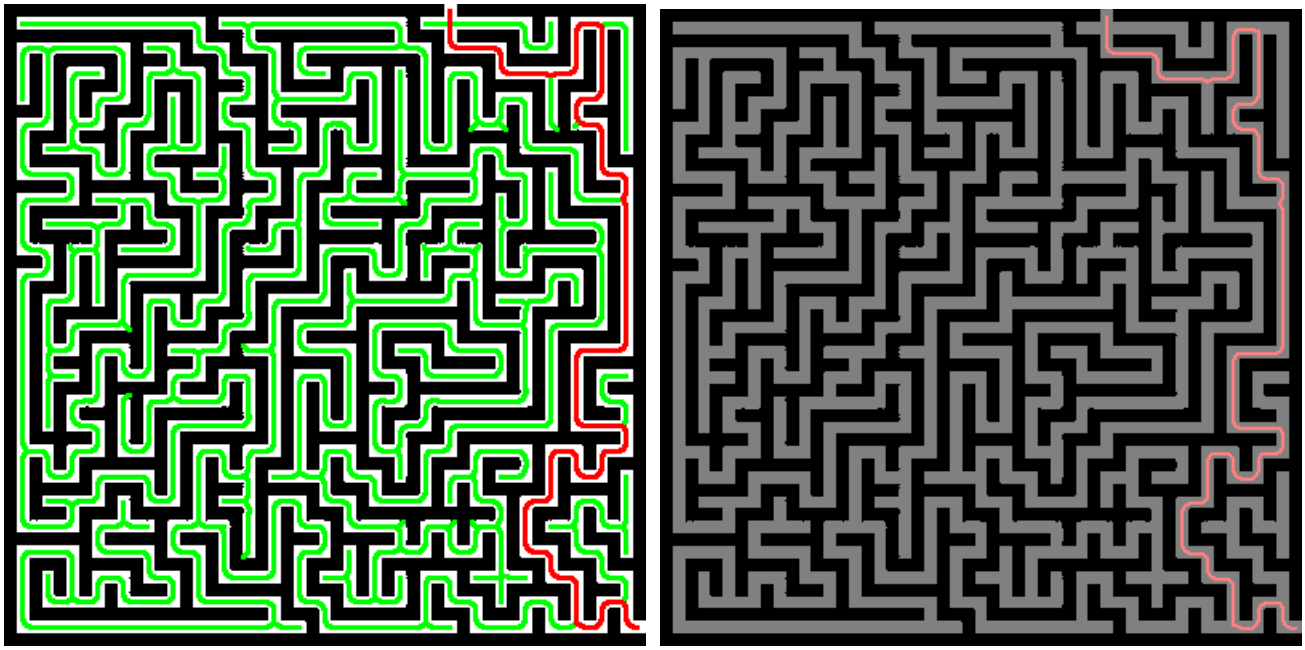
[1] Susanna S. Epp, Discrete mathematics with applications, Wadsworth, California, 1990.

[2] Y. Zhang and S. Waterman, "An Eulerian path approach to local multiple alignment for DNA sequences," *PNAS*, vol.102, no.5, pp.1285-1290, Feb. 2005.

[3] M. Matamala and E. Moreno, "Minimal Eulerian trail in a labeled digraph," *Tech. Report DIM-CMM*, Universidad de Chile, August 2004.

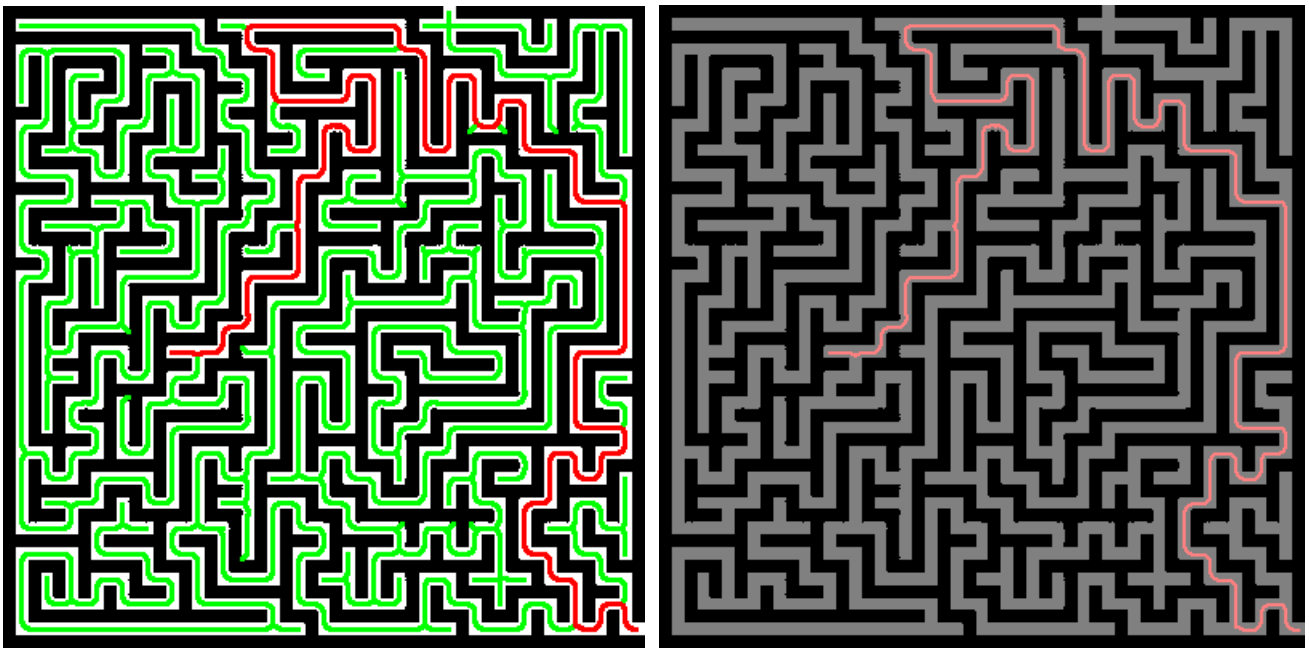
[4] Y. Qiao and M. Yasuhara, "Recovering drawing order from offline handwritten image using direction context and optimal Euler path," *ICASSP 2006*, vol.2, pp.765-768, 2006.

[5] A. Tucker, "A new applicable proof of the Euler circuit theorem," *The American Mathematical Monthly*, vol.83, no.8, pp.638-640, Oct. 1976.
[6] L.-H. Shih, and Y.-T. Lin, "Multicriteria optimization for infectious medical waste collection system planning," *Practice Periodical of Hazardous, Toxic, and Radioactive Waste Management*, ASCE, vol. 7, no. 2, pp.78-85, 2003.
[7] A. Mingozzi, "The multi-depot periodic vehicle routing problem," *SARA*, LNAI 3607,



(a) (b)

Fig.5. Resultant of Maze Game: (a) Euler path, and (b) final route of Fig.4(a). The green line represents the edge has been passed twice or more times; the red line is the final desired resultant.



(a) (b)

Fig.6. Resultant of Maze Game: (a) Euler path, and (b) final route of Fig.4(b). The green line represents the edge has been passed twice or more times; the red line is the final desired resultant.

pp.347-350, 2005.
[8] T. S. Ferguson, “Linear programming, a concise introduction, ”http://www.math.ucla.edu/~tom/LP.pdf.
[9] The Maze Generator, http://www.billsgames.com/mazegenerator/
[10] BlackDog’s Animal Mazes, http://blackdog4kids.com/games/maze/animal/index.html

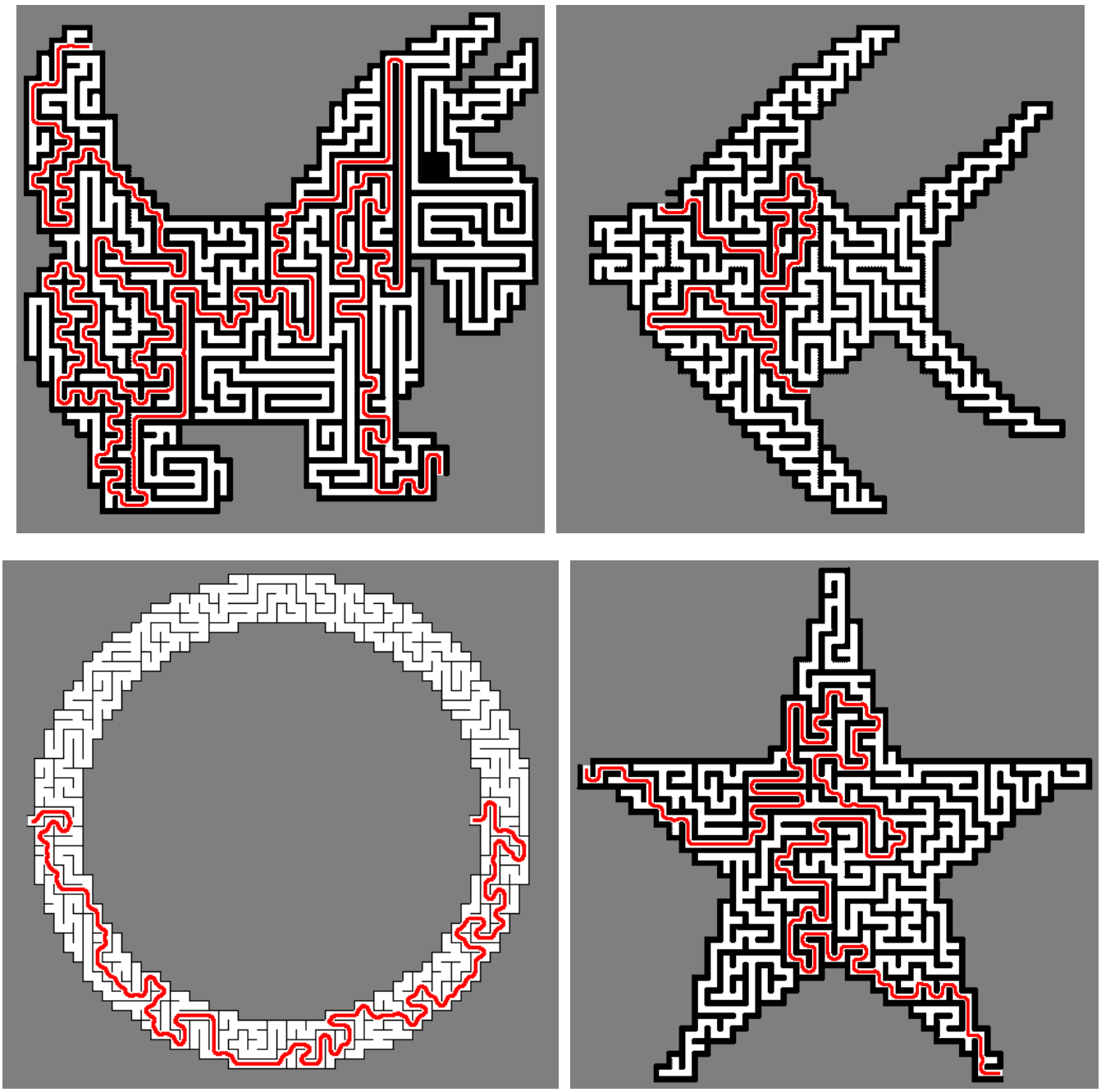


Fig.7. Resultants of four tested mazes [10].