𝕋ℂ𝕊-𝕋ℝ-𝔸-**14-79**

𝕋ℂ𝕊 Technical Report

Enumerating Eulerian Trails Based on Line Graph Conversion

by

Hiroyuki Hanada, Shuhei Denzumi, Yuma Inoue, Hiroshi Aoki, Norihito Yasuda, Shogo Takeuchi and Shin-ichi Minato

Division of Computer Science

Report Series A

October 9, 2014

Hokkaido University
Graduate School of
Information Science and Technology

Email: minato@ist.hokudai.ac.jp

Phone: +81-11-706-7682

Fax: +81-11-706-7682

Enumerating Eulerian Trails Based on Line Graph Conversion

Hiroyuki Hanada[†] Shuhei Denzumi[‡] Yuma Inoue[‡]
Hiroshi Aoki[‡] Norihito Yasuda[†] Shogo Takeuchi[†]
Shin-ichi Minato[†‡]
[†] ERATO Minato Project, Japan Science and Technology Agency
[‡] Graduate School of Information Science and Technology, Hokkaido University

October 9, 2014

(Abstract) Given an undirected graph G, we consider enumerating all Eulerian trails, that is, walks containing each of the edges in G just once. We consider achieving it with the enumeration of Hamiltonian paths with the zero-suppressed decision diagram (ZDD), a data structure that can efficiently store a family of sets satisfying given conditions. First we compute the *line graph* $L(G)$, the graph representing adjacency of the edges in G. We also formulated the condition when a path in $L(G)$ corresponds to a trail in G because every trail in G corresponds to a path in $L(G)$ but the converse is not true. Then we enumerate all Hamiltonian paths in $L(G)$ satisfying the condition with ZDD by representing them as their sets of edges.

1 Introduction

In the graph theory, an *Eulerian trail* of a graph G is a walk that contains each of the edges in G just once (definitions given in Sect. 2). It is known that we can easily judge whether a connected undirected graph G has an Eulerian trail: G has an Eulerian trail if and only if it has no or just two vertices of odd degree (Sect. 2). In addition, it is known that we can obtain an Eulerian trail of G in a simple manner called Fleury's algorithm [19]. However, it is considered difficult to enumerate *all* Eulerian trails: its time complexity is proved to be "#P-complete" (roughly speaking, time complexity "P-complete" for each trail) [3, 6, 15].

In this paper we consider efficient enumeration of Eulerian trails, especially by way of enumerating *Hamiltonian paths.* A Hamiltonian path of a graph G is a walk that contains each of the vertices in G just once. Although enumerating all Hamiltonian paths is not so easy either, many approaches have been proposed to enumerate all Hamiltonian paths [10, 13, 14, 17, 18], and we especially focus on the algorithm using *zero-suppressed binary decision diagram* (ZDD) [16] with the advantage described next.

We enumerate all Eulerian trails in a graph G as follows: First we compute the *line graph* $L(G)$ so that every Eulerian trail in a simple graph G correspond to a Hamiltonian path in $L(G)$. In addition, we discuss the condition when a path in $L(G)$ corresponds to a trail in G because not all paths in $L(G)$ correspond to trails in G (Sect. 3). Then we enumerate Hamiltonian paths in $L(G)$ satisfying the condition with the algorithm based on the *zero-suppressed binary decision diagram* (ZDD) [16]. ZDD efficiently store a family of sets, where we represent every path as a set of edges (Sect. 4.1). Another advantage of ZDD is that it can also efficiently exclude paths satisfying a given condition, which we use to exclude Hamiltonian paths in $L(G)$ not corresponding to Eulerian trails in G. Experimental results show that Eulerian trails in a given graph can be enumerated if the line graph has 120 edges or less, or more edges for certain type of graphs.

2 Definitions

We denote by (V, E) the graph whose set of vertices is V and whose set of edges is E, respectively. For an undirected graph G, it is called *simple* if there exists at most one edge between every pair of vertices and there does not exist any loop (an edge whose two ends are the same vertex).

For an undirected graph G, a pair of a sequence of vertices $(v_1, v_2, \ldots, v_m)$ and a sequence of edges $(e_1, e_2, \ldots, e_{m-1})$ is called a *walk* if the two ends of e_i are v_i and v_{i+1} for $i \in \{1, 2, \ldots, m-1\}$. We call a sequence of either vertices or edges also a walk if there exists the other sequence satisfying the condition above.

A walk is called to be *closed* if its sequence of vertices $(v_1, v_2, \ldots, v_m)$ satisfies $v_1 = v_m$. A *trail* is a walk whose edges in the sequence are distinct. A *path* is a walk whose vertices in the sequence are distinct (except for the precondition $v_1 = v_m$ if the walk is closed). Note that any path is also a trail. A closed path is called a *cycle*.

For a connected undirected graph G, it is called a *semi-Eulerian graph* (or an *Eulerian graph*) if there exists a trail (or a closed trail) containing all edges in G. Such a trail is called an *Eulerian trail*. Similarly, for a connected undirected graph G, it is called a *semi-Hamiltonian graph* (or a *Hamiltonian graph*) if there exists a path (or a cycle) containing all vertices in G. Such a path is called a *Hamiltonian path*. [19]

Let us review a well-known property for (semi-)Eulerian graphs: a connected undirected graph G is Eulerian if and only if the degrees of its vertices are all even, and G is semi-Eulerian if and only if the degrees of its vertices are all even except for just two vertices of odd degrees [7, 19].

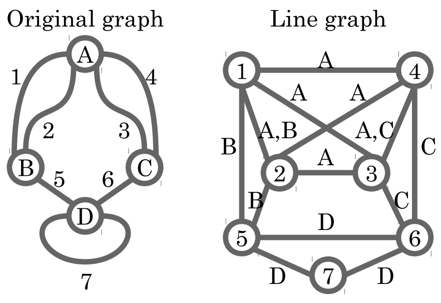


Figure 1: An example of a line graph. In the line graph in the figure, characters on edges represent the vertices in the original graph where the edges in the original graph are adjacent (see the "label" defined in Definition 2).

3 Representing Eulerian trails as Hamiltonian paths in the line graph

3.1 Line graph

Given a connected undirected simple graph G, we use its *line graph* $L(G)$ so that every Eulerian trail in G gives a Hamiltonian path in $L(G)$. The line graph $L(G)$ is a graph characterizing the adjacency of the edges in G as follows: $L(G)$ has vertices corresponding one-to-one to the edges in G, and there exists an edge in $L(G)$ between the two vertices u and v if and only if two edges corresponding to u and v are adjacent in G [4, 7]. An example is shown in Fig. 1. In this paper we define it formally as follows:

Definition 1. *Given two undirected graphs $G = (V, E)$ and $G' = (V', E')$, G' is called the* line graph *of G, denoted by $L(G)$, if*

- *V' corresponds to E one-to-one, that is, there exists a bijection $l : E \to V'$, and*
- *For any $v'_1, v'_2 \in V'$, E' has an edge between v'_1 and v'_2 if and only if $l^{-1}(v'_1), l^{-1}(v'_2) \in E$ are adjacent in G.*

It is easily proved that, given a trail t in G, traversing the vertices in $L(G)$ in the order of corresponding edges in t gives a path in $L(G)$. An example is shown in Fig. 2.

Property 1. *Let $G = (V, E)$, $L(G) = (V', E')$, and t be a trail in G whose sequence of edges is $(e_1, e_2, \ldots, e_m)$ $(e_i \in E)$. Then the sequence of vertices in $L(G)$: $(l(e_1), l(e_2), \ldots, l(e_m))$ $(l(e_i) \in V')$ is a path in $L(G)$.*

We extend the definition of l: for a trail t in G, we denote by $l(t)$ the path above in $L(G)$.

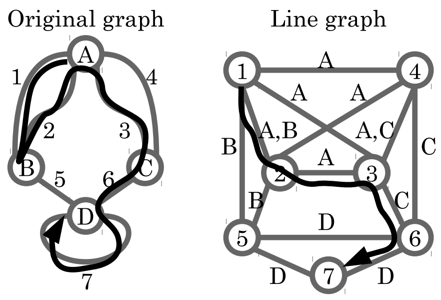


Figure 2: An example of a trail in a graph and the corresponding path in the line graph.

Proof. Because t is a trail, for any $i \in \{1, 2, \dots, m-1\}$, two edges e_i and e_{i+1} are adjacent and therefore two vertices $l(e_i)$ and $l(e_{i+1})$ are connected from the definition of the line graph. Thus $(l(e_1), l(e_2), \dots, l(e_m))$ is a walk in $L(G)$. In addition, because $e_1, e_2, \dots, e_m$ are distinct from the fact that t is a trail, $l(e_1), l(e_2), \dots, l(e_m)$ are also distinct. (Recall that l is a bijection.) This means $(l(e_1), l(e_2), \dots, l(e_m))$ is a path in $L(G)$. ∎

In Property 1, if t contains all edges in G, then $l(t)$ contains all vertices in $L(G)$. As a result,

Property 2. *Let t be a Eulerian trail in a (semi-)Eulerian graph G. Then $l(t)$ is a Hamiltonian path in $L(G)$. (This implies $L(G)$ is a (semi-)Hamiltonian graph.)*

The fact that $L(G)$ is a (semi-)Hamiltonian graph if G is a (semi-)Eulerian graph is already discussed in 1960s [2, 8]. Especially, the condition when $L(G)$ is Hamiltonian is described as follows:

Property 3. *[2] For a graph $G = (V, E)$, $L(G)$ is Hamiltonian if and only if G is* sequential*, where G is called sequential if there exists a permutation of E: $(e_1, e_2, \dots, e_m)$ ($e_i \in E$, $m = |E|$) such that e_i and e_{i+1} are adjacent for all $i \in \{1, 2, \dots, m\}$ (treat $e_{m+1} = e_1$).*

Note that, although the result above is for Hamiltonian cycles and closed Eulerian trails, it is easily extended for non-closed ones.

G is sequential if G is Eulerian but the converse does not always hold. As a result, enumerating all Hamiltonian paths in $L(G)$ is excessive for representing all Eulerian trails in G. Thus, to enumerate Eulerian trails in G by the Hamiltonian paths in $L(G)$, we have only to exclude such excessive paths. However, the condition of exclusion has not been clarified as far as the authors know, although it is known in 1963 at latest that no exclusion is needed if G is directed[1], that is, Hamiltonian paths in $L(G)$ correspond one-to-one to Eulerian trails in G in case G is directed [10]. In the next section we clarify the condition for undirected graphs, although it may be already proved by others.

[1]In this paper we omit the definition of the line graph of a directed graph. See the reference.

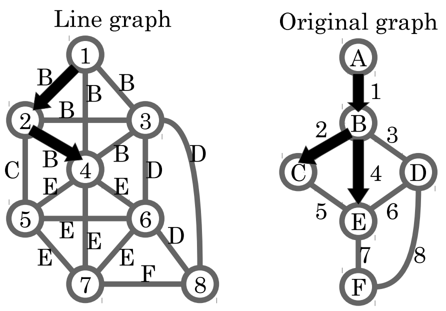


Figure 3: An example of a path in a line graph of a simple graph that does not have a corresponding trail in the original graph.

3.2 The condition when a path in a line graph represents a trail in the original graph

Let us assume G is a connected undirected simple graph and consider when a path p in $L(G)$ does not correspond to an trail in G (not limited to Hamiltonian and Eulerian). An example is shown in Fig. 3. In this case, three successive edges in G shares a vertex and thus it is not a trail in G.

The following discussion assures only such a case makes a path in $L(G)$ does not correspond to an trail in G if G is simple. First, to characterize the edges in $L(G)$, we define their *labels* as follows:

Definition 2. *For a simple graph G and an edge $e' = (u', v')$ ($e \in E'$, $u, v \in V'$) in the line graph $L(G) = (V', E')$, we define the* label *of e', denoted by $\lambda(e')$, by the only vertex in G where the two edges in G: $l^{-1}(u')$ and $l^{-1}(v')$ are adjacent.*

Note that, because no two edges in a simple graph G exist between the same pair of vertices, the vertex for the label must be unique. In case G is not simple, an edge in $L(G)$ may have two labels (e.g. Fig. 1 in Sect. 3.1).

From the definition of the label, in case three successive edges in G share a vertex like in Fig. 3, the labels in the corresponding two edges in $L(G)$ must be the same. We can formulate the fact as follows:

Theorem 1. *Let $G = (V, E)$ be a connected undirected simple graph and p be a path in $L(G) = (V', E')$. Then the followings are equivalent: (A) there exists a trail t in G such that $p = l(t)$ and (B) the same edge label does not appear successively in the sequence of edges for p.*

Proof. Let p has a subpath $v'_1 \overset{e'_1}{\rightarrow} v'_2 \overset{e'_2}{\rightarrow} v'_3$ ($v'_i \in V'$, $e'_i \in E'$). Then, $l^{-1}(v'_1), l^{-1}(v'_2), l^{-1}(v'_3) \in E$ are connected in G as follows:

(X) In case the labels of e'_1 and e'_2 are the same, the three edges $l^{-1}(v'_1), l^{-1}(v'_2), l^{-1}(v'_3) \in E$ shares the vertex of the label. Thus these three edges are connected at a vertex in G (Case 1 in Fig. 4).

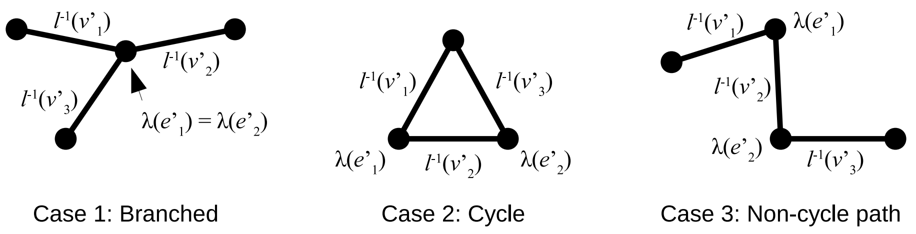


Figure 4: All possible cases of three edges in a simple graph.

(Y) In case the labels of e'_1 and e'_2 are different,

(Y1) If the two edges $l^{-1}(v'_1)$ and $l^{-1}(v'_3)$ are adjacent in G, then there exists an edge between v'_1 and v'_3 in $L(G)$, where its label is different from the other two. Thus the three edges yield a cycle (Case 2 in Fig. 4).

(Y2) If the two edges $l^{-1}(v'_1)$ and $l^{-1}(v'_3)$ are not adjacent in G, then they yields a non-cycle path (Case 3 in Fig. 4).

From the consideration, we prove (A) and (B) are equivalent.

Proof of (A) ⇒ (B): Suppose p, a path in $L(G)$, has two successive edges with the same label, that is, there exist a subpath $v'_1 \overset{e'_1}{\to} v'_2 \overset{e'_2}{\to} v'_3$ in $L(G)$ with $\lambda(e'_1) = \lambda(e'_2)$. In this case $l^{-1}(v'_1)$, $l^{-1}(v'_2)$ and $l^{-1}(v'_3)$, three edges in G, must be adjacent with the form of (X) among (X), (Y1) and (Y2) above. This contradicts the precondition that $(l^{-1}(v'_1), l^{-1}(v'_2), l^{-1}(v'_3))$ is a subtrail in G.

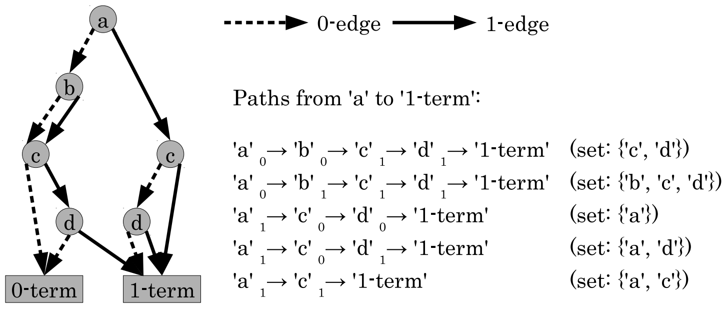
Proof of (B) ⇒ (A): Let p be a path in $L(G)$ without any two successive edges with the same label. Then, for any three successive vertices in p, corresponding three edges in G must take the form of (Y1) or (Y2). This implies no branching edges exist in the sequence of edges and thus the whole p corresponds to a trail in G.

∎

4 Enumerating Hamiltonian paths in the line graph corresponding to Eulerian trails

4.1 Representing Hamiltonian paths by zero-suppressed binary decision diagram

As an algorithm of enumerating Hamiltonian paths satisfying given conditions, we use the method based on the *zero-suppressed binary decision diagram* (ZDD) [16],



Paths from 'a' to '1-term':

'a' $\xrightarrow[0]{}$ 'b' $\xrightarrow[0]{}$ 'c' $\xrightarrow[1]{}$ 'd' $\xrightarrow[1]{}$ '1-term' (set: {'c', 'd'})
'a' $\xrightarrow[0]{}$ 'b' $\xrightarrow[1]{}$ 'c' $\xrightarrow[1]{}$ 'd' $\xrightarrow[1]{}$ '1-term' (set: {'b', 'c', 'd'})
'a' $\xrightarrow[1]{}$ 'c' $\xrightarrow[0]{}$ 'd' $\xrightarrow[0]{}$ '1-term' (set: {'a'})
'a' $\xrightarrow[1]{}$ 'c' $\xrightarrow[0]{}$ 'd' $\xrightarrow[1]{}$ '1-term' (set: {'a', 'd'})
'a' $\xrightarrow[1]{}$ 'c' $\xrightarrow[1]{}$ '1-term' (set: {'a', 'c'})

Figure 5: An example of ZDD. This ZDD represents the family of five sets {'c', 'd'}, {'b', 'c', 'd'}, {'a'}, {'a', 'd'} and {'a', 'c'}.

a data structure originally for representing binary functions and also for storing families of sets. A famous algorithm for the enumeration with ZDD is proposed by Knuth [12], called *SIMPATH* in his website [11]. First we show the outline of ZDD.

Definition 3. *[16] Given a sequence of boolean variables* $A = (a_1, a_2, \ldots, a_n) : \{0,1\}^n$ *and a boolean function* $f(a_1, a_2, \ldots, a_n) : \{0,1\}^n \to \{0,1\}$*,* zero-suppressed binary decision diagram *(ZDD) for* f *is a minimal directed acyclic graph (DAG) such that:*

- *Every vertex is named one of the followings: an element of* A*, "0-terminal" or "1-terminal". There exists just one vertex named* a_1*, "0-terminal" or "1-terminal".*
- *Every vertex has two outgoing edges named "0-edge" and "1-edge". No edge connects from* a_i *to* a_j *with* $i \geq j$*.*
- $f(a_1, a_2, \ldots, a_n)$ *takes 1 for the arguments defined by paths from* a_1 *to "1-terminal" in the diagram as follows: for every path above,* a_i $(i = 1, 2, \ldots, n)$ *takes 1 if there exists a vertex named* a_i *which is a source of "1-edge" in the path, otherwise* a_i *takes 0. For the other arguments* $f(a_1, a_2, \ldots, a_n)$ *takes 0.*

ZDD can represent a family of sets by regarding A as the universal set, the assignments for variables $a_1, a_2, \ldots, a_n$ as the existence of the elements in a set, and the function value $f(a_1, a_2, \ldots, a_n)$ as taking 1 if the set is contained in the family or 0 otherwise. ZDD is invented as a variant of BDD (binary decision diagram) [1] so that the diagram becomes smaller when f takes zero for most of the elements in A, that is, the number of sets stored in the family is much fewer than 2^n (the number of all possible sets). An example is shown in Fig. 5.

As stated in the definition, ZDD must be minimal, that is, the nodes in ZDD must be removed or merged as long as the resulted binary function is not changed. Concretely, we apply the operations in Fig. 6 to make the ZDD minimal [16].

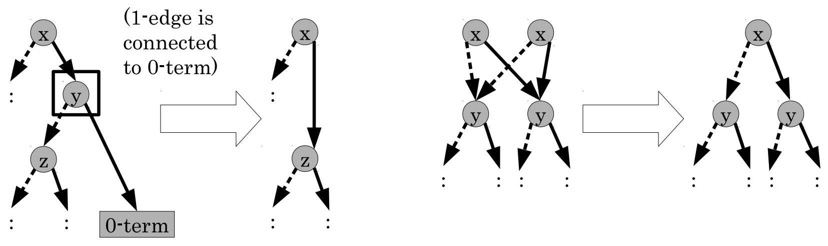


Figure 6: The reduction rules of ZDD [16]. The first one is to remove excessive node: in case there is a node whose 1-edge is connected to 0-term, remove the node and connect its parent to its destination of 0-edge. The second one is to merge two nodes contributing to the same binary function.

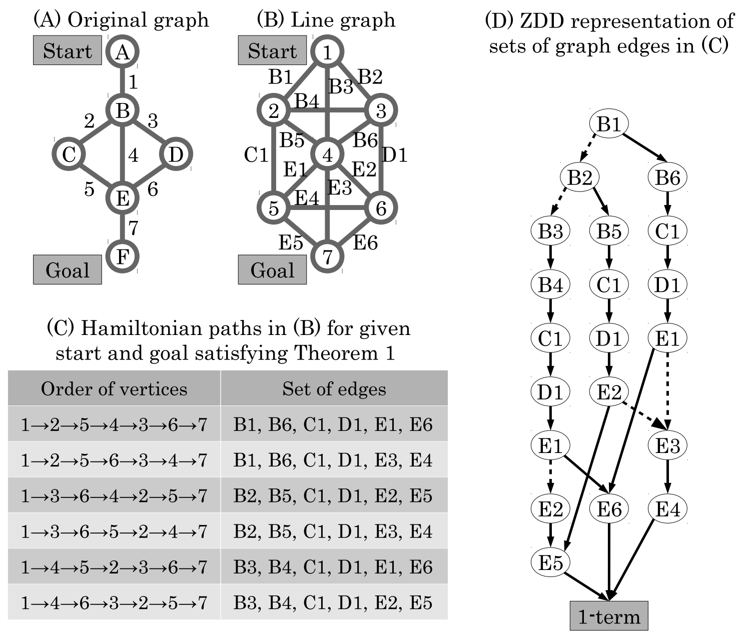
We use a ZDD whose universal set is the edges in $L(G)$, and let the family of edges represent the Hamiltonian paths in $L(G)$ corresponding to Eulerian paths in G (satisfying Theorem 1). Because paths, different from trails, are distinguished by the set of edges (not *sequences* of edges), paths are much easier to be represented by ZDD than trails. In addition, it is also efficient to apply set operations like "excluding sets (paths) containing certain elements (edges)" for the sets in ZDD, applying the condition of Theorem 1 is expected to be easy. An example is shown in Fig. 7.

4.2 Algorithm for enumerating Eulerian trails

Summarizing the discussions, the following algorithm enumerates all Eulerian trails in a connected undirected (semi-)Eulerian graph G.

1. Make G simple without changing the number of the Eulerian trails in it so that Theorem 1 can be applied. Precisely,
   - In case there exist a pair of vertices with two or more edges between them, split each of the edges into two by inserting a vertex except for arbitrary one edge.
   - In case there exist a loop edge (Sect. 2), split it into three by inserting two vertices[2].
2. Add some vertices and edges to G so that the start and goal vertices of Hamiltonian paths in $L(G)$ is made unique, without changing the number of the Eulerian trails in it. Precisely,
   - In case G is semi-Eulerian, for each of the two vertices with odd degree, create a dummy vertex and an edge to connect to the vertex of odd

[2]In this case we treat two ends of the loop edge are distinguished: for example, we distinguish the two cases of traversing the edge '7' in Fig. 1 clockwisely or counterclockwisely. The algorithm for treating them not distinguished is not developed yet.



(C) Hamiltonian paths in (B) for given start and goal satisfying Theorem 1

Order of vertices	Set of edges
1→2→5→4→3→6→7	B1, B6, C1, D1, E1, E6
1→2→5→6→3→4→7	B1, B6, C1, D1, E3, E4
1→3→6→4→2→5→7	B2, B5, C1, D1, E2, E5
1→3→6→5→2→4→7	B2, B5, C1, D1, E3, E4
1→4→5→2→3→6→7	B3, B4, C1, D1, E1, E6
1→4→6→3→2→5→7	B3, B4, C1, D1, E2, E5

Figure 7: An example of ZDD representation of Hamiltonian paths satisfying Theorem 1 in a line graph. In the figure of (D), unspecified ZDD edges are regarded as being connected to 0-terminal. (For example, the destination of 0-edge for ‘B6’ is 0-terminal.

degree unless the degree is 1. This assures the start and goal edges of the Eulerian trails in G being unique, that is, the start and goal vertices of the Hamiltonian paths in $L(G)$ being unique (See Fig. 8).

- In case G is Eulerian, determine a start and goal (the same) vertex in G, then create two dummy vertices and two edges connected to the vertex. (The set of trails may change by the vertex.) As a result, G becomes semi-Eulerian.

3. Create $L(G)$ from G. Simultaneously, classify all edges in $L(G)$ by their labels.

4. Construct the constraint of Theorem 1 given to ZDD. Since the constraint must be given by sets of edges rather than sequences of the edges (Sect. 4.1), we describe it as follows: “For every pair of adjacent edges in $L(G)$ with the same label, they does not appear simultaneously in a path.”

5. Enumerate all Hamiltonian paths in $L(G)$ satisfying the constraint for the start and goal edges with the SIMPATH method.

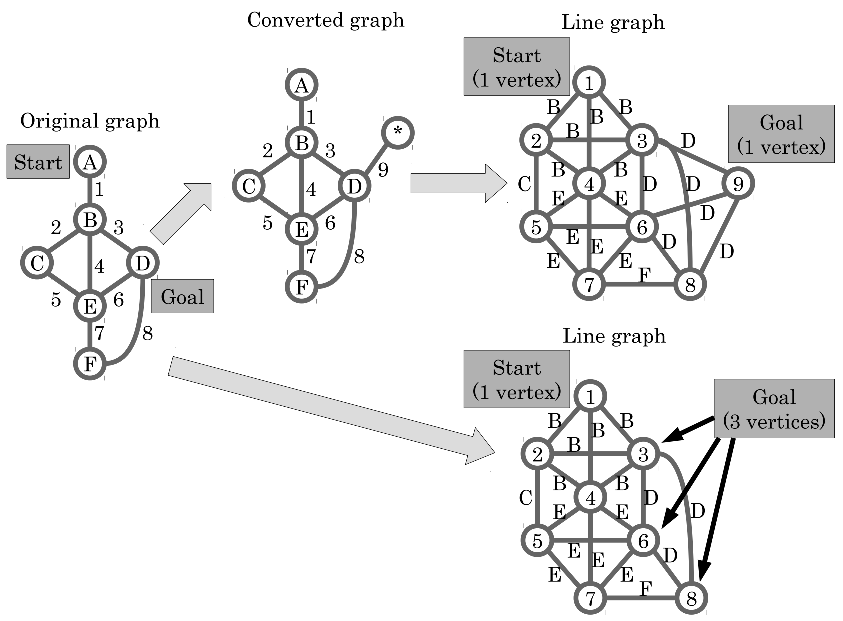


Figure 8: An example of a vertex and an edge additions for a semi-Eulerian graph to assure the start/goal vertices in $L(G)$ made unique. In this case, because the degree of the vertex ‘D’ is 3, an odd number larger than 1, we add a dummy vertex ‘*’ and an edge ‘9’.

5 Experiment

5.1 Setting

We implemented the algorithm of Sect. 4.2 with Graphillion [9], a Python library for graphs and their paths based on ZDD in the manner in Sect. 4.1.

We enumerated the Eulerian trails in the four types of graphs in Tables 1 and 2. Because the line graph $L(G)$ has $d(d-1)/2$ edges for each vertex of degree d in G, the time and space for the computation are expected to grow much for increasing vertex degree even if the number of edges in G is not so increased. Thus we experimented graphs with constant maximum degree (Ring, Diamond) and increasing degree (Bunch, Complete). As seen in Table 2, the number of edges in Bunch and Complete are multiplied by $\Theta(k)$ after the conversions to the line graphs.

We measured the computation times of the enumeration; times for setting up graph structures (converting given graphs to their line graphs, adding dummy vertices for making them simple and unique the start and goal vertices) and obtaining paths which are Hamiltonian and satisfying the condition of Theorem 1. The experiment was conducted on a computer with the CPU “AMD A4-5000 APU” (clock: 1.5GHz) and 4GB RAM. The running time for each graph is limited to one hour.

Table 1: Graphs examined in the experiment

Name	Structure	#Vertices	#Edges	Maximum degree
Ring(k)	Fig. 9	$2k+2$	$4k+1$	4
Diamond(k)	Fig. 10	$2k(k+1)+2$	$4k^2+2$	4
Bunch(k)	Fig. 11	2	k	k
Complete(k)	Fig. 12	k	$\frac{k(k-1)}{2}$	$k-1$

Table 2: The properties of the graphs after making the graph simple and adding dummy vertices (Sect. 4.2).

Name	#Vertices	#Edges	#LineGraphEdges	Maximum degree
Ring(k)	$3k+3$	$5k+2$	$13k+1$	4
Diamond(k)	$2k(k+1)+2$	$4k^2+2$	$4k(3k-2)+9$ †	4
Bunch(k)	$k+3$	$2k+1$	$k(k+2)$ ‡	$k+2$ ‡
Complete(k)	$k+2$	$\frac{k(k-1)}{2}+2$	$\frac{k^3-3k^2+6k-2}{2}$	$k-1$

† The value is exceptional for $k = 1$. (Number of edges is 9 in the line graph of Diamond(1).)

‡ The values become smaller by 1 for odd k.

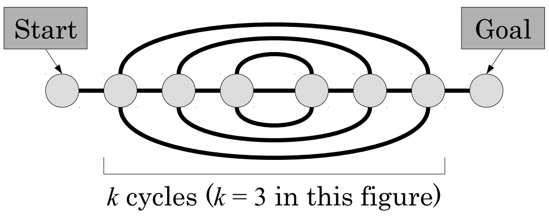


Figure 9: The graph Ring(k) ($k = 3$)

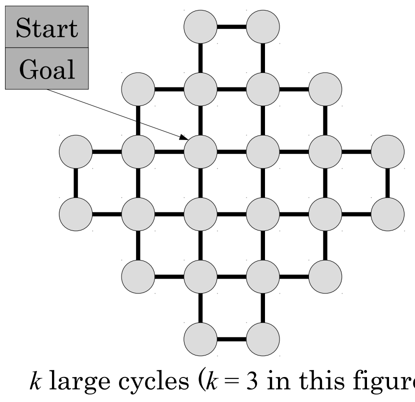


Figure 10: The graph Diamond(k) ($k = 3$). The start and the goal is set at the one of the four central vertices. This graph is a variant of *Aztec diamond* [5].

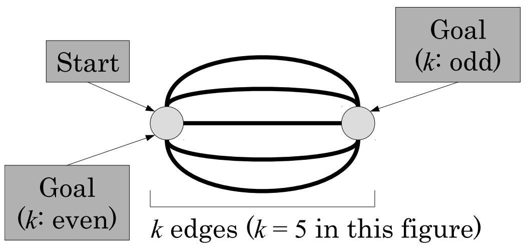


Figure 11: The graph Bunch(k) ($k = 5$)

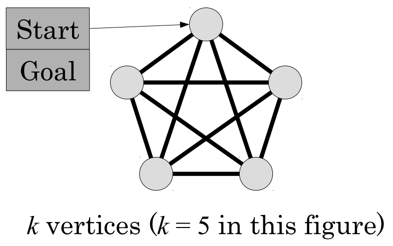


Figure 12: The graph Complete(k) ($k = 5$). Such a graph is called *complete graph* [4,7,19]. It has Eulerian trails if and only if k is odd.

5.2 Result

We show the results of computation times and numbers of trails in Table 3. Table 4 describes the numbers of vertices and edges in the line graph.

From Table 4 with the three graphs Diamond(k), Bunch(k) and Complete(k), it seems to be possible to enumerate Eulerian trails if there are about 120 edges or less in the line graph. However, as shown by Ring(k), more edges would be acceptable according to the shape of graphs. It is easily assumed, but yet to be examined, that the Hamiltonian paths in $L(\text{Ring}(k))$ satisfying Theorem 1 are well compressed by ZDD.

As for the computation times, they grow rapidly for increasing k except for Ring(k) in almost linear against k. The time for Ring(k) is unexpectedly fast because, in general, we need $O(2^n)$ time and space to compute ZDD for a universal set of size n.

There remains the problem of what parameter is essential for fast computation. From the property of ZDD, it is clear that the number of edges affects much. However, we should examine other parameters from the result of Ring(k): in fact, Eulerian trails in Ring(20) (102 vertices and 261 edges in the line graph) is much easier to be computed than that in Diamond(5) (102 vertices and 269 edges in the line graph).

Table 3: Number of Eulerian trails and computation times (sec) of four types of graphs

	Ring(k)		Diamond(k)		Bunch(k)		Complete(k)	
k	#trails	time	#trails	time	#trails	time	#trails	time
1	6	0.0089	2	0.0073	1	0.0055	—	
2	36	0.0131	160	0.0171	2	0.0070	—	
3	216	0.0170	528640	0.0602	6	0.0086	2	0.0076
4	1296	0.0209	1.3×10^{11}	7.9481	24	0.0129	—	
5	7776	0.0270	Memory out		120	0.0190	528	0.0214
6	46656	0.0306			720	0.0447	—	
7	279936	0.0322			5040	0.0515	3.8×10^{8}	94.1611
8	1679616	0.0398			40320	4.4528	—	
9	10077696	0.0435			362880	0.4301	Time out	
10	6.0×10^{7}	0.0458			3628800	6.9748		
11	3.6×10^{8}	0.0549			39916800	11.2032		
12	2.1×10^{9}	0.0560			Memory out			
13	1.3×10^{10}	0.0625						
14	7.8×10^{10}	0.0653						
15	4.7×10^{11}	0.0675						
20	3.6×10^{15}	0.1027						
30	2.2×10^{23}	0.1698						

Table 4: Numbers of vertices and edges in the line graph. Note that the numbers of vertices and edges are equivalent to #Edges and #LineGraphEdges in Table 2, respectively. Underlined numbers denote the cases of failed computations (memory-out or time-out).

	$L(\text{Ring}(k))$		$L(\text{Diamond}(k))$		$L(\text{Bunch}(k))$		$L(\text{Complete}(k))$	
k	vertices	edges	vertices	edges	vertices	edges	vertices	edges
1	7	14	6	9	3	2	—	
2	12	27	18	41	5	8	—	
3	17	40	38	93	7	14	5	8
4	22	53	66	169	9	24	—	
5	27	66	$\underline{102}$	$\underline{269}$	11	34	12	39
6	32	79			13	48	—	
7	37	92			15	62	23	118
8	42	105			17	80	—	
9	47	118			19	98	$\underline{38}$	$\underline{269}$
10	52	131			21	120		
11	57	144			23	142		
12	62	157			$\underline{25}$	$\underline{168}$		
13	67	170						
14	72	183						
15	77	196						
⋮								
20	102	261						
30	152	391						

6 Conclusion

In this research we considered enumerating all Eulerian trails in an undirected graph G. We focus on ZDD-based Hamiltonian path enumeration, which can enumerate not only all Hamiltonian paths but also Hamiltonian paths satisfying certain conditions. We consider converting G into the line graph $L(G)$, where every Eulerian trail in G corresponds to a Hamiltonian path in $L(G)$ (Sect. 3.1). In addition, because not all Hamiltonian paths in $L(G)$ corresponds to Eulerian trails in $L(G)$, we formulated the condition by defining "labels" of the edges in $L(G)$ (Theorem 1 in Sect. 3.2). As a result of the experiment, we could enumerate Eulerian trails in G if $L(G)$ has 120 or less edges, although more edges can be accepted for certain type of graphs.

We consider the following problems as future works: finding parameters of graphs determining the computational time other than the number of vertices and edges, and developing more memory-efficient data structure.

References

[1] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[2] G. Chartrand. On Hamiltonian line-graphs. *Transactions of the American Mathematical Society*, 134:559–566, 1968.

[3] P. Creed. Sampling Eulerian orientations of triangular lattice graphs. *Journal of Discrete Algorithms*, 7(2):168–180, 2009.

[4] R. Diestel. *Graph Theory.* Springer, 4th edition, 2010.

[5] N. Elkies, G. Kuperberg, M. Larsen, and J. Propp. Alternating-sign matrices and domino tilings (part I). *Journal of Algebraic Combinatorics*, 1(2):111–132, 1992.

[6] Q. Ge and D. Štefankovič. The complexity of counting Eulerian tours in 4-regular graphs. *Algorithmica*, 63(3):588–601, 2012.

[7] F. Harary. *Graph Theory.* Addison-Wesley, 1st edition, 1969.

[8] F. Harary and C. S. J. A. Nash-Williams. On Eulerian and Hamiltonian graphs and line graphs. *Canadian Mathematical Bulletin*, 8:701–709, 1965.

[9] T. Inoue, H. Iwashita, J. Kawahara, and S. Minato. Graphillion: Software library designed for very large sets of graphs in python. Technical Report TCS-TR-A-13-65, 2013.

[10] P. W. Kasteleyn. A soluble self-avoiding walk problem. *Physica*, 29(12):1329–1337, 1963.

[11] D. E. Knuth. Don Knuth's home page. `http://www-cs-staff.stanford.edu/~uno/`.

[12] D. E. Knuth. *Combinatorial Algorithms*, volume 4A of *The Art of Computer Programming*, chapter 7.1.4 Binary Decision Diagrams. Pearson Education, 2011.

[13] H. Liu and J. Wang. A new way to enumerate cycles in graph. In *International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications*, page 57, Feb 2006.

[14] P. Mateti and N. Deo. On algorithms for enumerating all circuits of a graph. *SIAM Journal on Computing*, 5(1):90–99, 1976.

[15] M. Mihail and P. Winkler. On the number of Eulerian orientations of a graph. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '92, pages 138–145, 1992.

[16] S. Minato. Zero-suppressed BDDs and their applications. *International Journal on Software Tools for Technology Transfer*, 3(2):156–170, 2001.

[17] F. Rubin. A search procedure for Hamilton paths and circuits. *Journal of the ACM*, 21(4):576–580, Oct. 1974.

[18] N. J. van der Zijpp and S. F. Catalano. Path enumeration by finding the constrained k-shortest paths. *Transportation Research Part B: Methodological*, 39(6):545–563, 2005.

[19] R. J. Wilson. *Introduction to Graph Theory.* Pearson Education, 4th edition, 1996.