Minimal Eulerian Circuit in a Labeled Digraph*

Eduardo Moreno and Martín Matamala

Departamento de Ingeniería Matemática, Centro de Modelamiento Matemático,
Universidad de Chile, UMR 2071, UCHILE-CNRS,
Casilla 170-3, Correo 3, Santiago, Chile
emoreno@dim.uchile.cl, mmatamal@dim.uchile.cl

Abstract. Let $G = (V, A)$ be an Eulerian directed graph with an arc-labeling. In this work we study the problem of finding an Eulerian circuit of lexicographically minimal label among all Eulerian circuits of the graph. We prove that this problem is NP-hard by showing a reduction from the DIRECTED-HAMILTONIAN-CIRCUIT problem.

If the labeling of the arcs is such that arcs going out from the same vertex have different labels, the problem can be solved in polynomial time. We present an algorithm to construct the unique Eulerian circuit of lexicographically minimal label starting at a fixed vertex. Our algorithm is a recursive greedy algorithm which runs in $\mathcal{O}(|A|)$ steps.

We also show an application of this algorithm to construct the minimal De Bruijn sequence of a language.

1 Introduction

Eulerian graphs were an important concept in the beginning of graph theory. The "Königsberg bridge problem" and its solution given by Euler in 1736 is considered the first paper of what is nowadays called *graph theory*.

In this work, we consider Eulerian digraphs with an arc-labeling into a finite alphabet, and we study the problem of finding the Eulerian circuit of lexicographically minimal label among all Eulerian circuits in the digraph.

By the BEST theorem (see [1]), we can compute the number of Eulerian circuits in a graph. This number is usually exponential in the number of vertices of the graph (at least $((\gamma - 1)!)^{|V|}$ where V is the set of vertices and γ is the minimum degree of vertices in V). Therefore, finding the Eulerian circuit of lexicographically minimal label can be costly.

This problem can be stated as a Chinese postman problem with a kind of priority over the streets: The postman must deliver mail in a network of streets and return to his depot without walking any street more than once (minimizing the walked distance) and at each corner he wants to choose the street of minimal slope. Therefore, the post office needs to give an itinerary to the postman such that at each corner he will choose the unvisited street of minimal slope unless it produces an unfeasible itinerary.

* Partially supported by Programa Iniciativa Científica Milenio P01-005 and Fundación Andes.

J.R. Correa, A. Hevia, and M. Kiwi (Eds.): LATIN 2006, LNCS 3887, pp. 737–744, 2006.

To find an Eulerian circuit of lexicographically minimal label is also interesting with respect to the problem of finding optimal encodings for DRAM address bus. In this model, an address space of size 2^{2n} is represented as labels of arcs in a complete digraph with 2^n vertices. An Eulerian circuit over this digraph produces an optimal multiplexed code (see [2]). If we want to give priority to some address in particular, an Eulerian circuit of lexicographically minimal label give us this code.

Eulerian digraphs with an arc-labeling are commonly employed in automata theory: a labeled digraph represents deterministic automata where vertices are the states of the automata, and arcs represent the transitions from one state to another, depending on the label of the arc. Eulerian circuits over these digraphs are related with synchronization of automata (see [3]).

Eulerian digraphs with an arc-labeling are also used in the study of DNA. By DNA sequencing we can obtain fragments of DNA which need to be assembled in the correct way. To solve this problem, we can simply construct a *DNA graph* (see [4]) and find an Eulerian circuit over this digraph. This strategy is already implemented and it is now one of the most promising algorithms for DNA sequencing (see [5, 6]).

Another interesting application of these digraphs is to find *De Bruijn sequences* of a language. De Bruijn sequences are also known as "shift register sequences" and were originally studied in [7] by N. G. De Bruijn for the binary alphabet. These sequences have many different applications, such as memory wheels in computers and other technological device, network models, DNA algorithms, pseudo-random number generation and modern public-key cryptographic schemes, to mention a few (see [8, 9, 10]). More details about this application are discussed in Section 3.

Note that these last applications consider digraphs with an arc-labeling with a particular property: Arcs going out from the same vertex have different labels.

In Section 2, we define the problem and we study its complexity. We prove that the problem is NP-hard. In Section 3 we study the problem when the arc-labeling has different labels for arcs going out from the same vertex. We show that in this case the problem can be polynomially solved: we give a recursive greedy algorithm that runs in linear time in the number of arcs of the digraph. Finally, in Section 4 we show an application of this algorithm to construct the minimal De Bruijn sequence of a language.

2 The Problem and Its Complexity

Let G be a digraph and let $l : A(G) \to N$ be a labeling of the arcs of G over an alphabet N such that arcs going out from the same vertex have different labels.

A *trail* is a sequence $T = a_1 a_2 \dots a_k$ of arcs a_j such that the tail of a_i is the head of a_{i-1} for every $i = 1, 2, \dots, k$ and all arcs are distinct. If the tail of a_1 is equal to the head of a_k then T is a closed trail or *circuit*. A circuit is an Eulerian circuit if the arcs of T are all the arcs of G. An Eulerian digraph is a digraph with an Eulerian circuit. The label of T, $l(T)$, is the word $l(a_1) \dots l(a_k)$.

Our problem is the following: given an Eulerian digraph and a vertex r, we intend to find the Eulerian circuit starting in r with the lexicographically minimal label. We note that is important to fix a starting vertex r so as to define an order in which vertices are visited, which allows us to define a lexicographical order among Eulerian circuits.

First, we prove that this problem is NP-hard. We define the decision problem:

MIN-LEX-EULERIAN-CIRCUIT
Instance: An Eulerian digraph G, a labeling $l : A(G) \to N$ of its arcs, a starting vertex r and a word $X \in N^{|A(G)|}$.
Question: Is there an Eulerian circuit T starting at r such that $l(T) \leq X$?

Theorem 1. MIN-LEX-EULERIAN-CIRCUIT *is NP-complete.*

Proof. We present a transformation of a DIRECTED-HAMILTONIAN-CIRCUIT instance (see [11]) into a MIN-LEX-EULERIAN-CIRCUIT instance, polynomially bounded in the size of the input graph.

Let G be a digraph. We want to verify if G contains a directed Hamiltonian circuit. We construct a digraph H in the following way: for each vertex $v \in G$, we include two vertices v_1 and v_2 and an arc v_1v_2 in H. Additionally, for each arc $vw \in G$ we include the arc v_2w_1 in H (see Figure 1). Finally, we label all arcs in H with the label 0.

It is easy to see that G has a Hamiltonian circuit if and only if H has a circuit with label $0^{2|V(G)|}$.

We can complete the digraph H to an Eulerian digraph $\bar{H}$ with additional vertices and arcs with label 1 in the following way: we add a vertex c and we

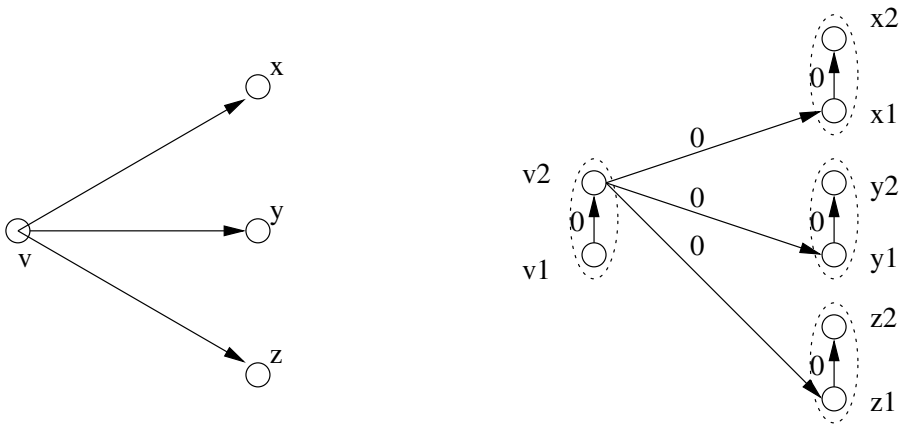

Fig. 1. Transformation of a digraph G (DIRECTED-HAMILTONIAN-CIRCUIT instance) into a labeled digraph H (MIN-LEX-EULERIAN-CIRCUIT instance)

connect every vertex v in H to c with two arcs vc and cv of label 1. With these connections, the resulting digraph is strongly connected even if we remove all arcs in H. Finally, to each arc xy in H we add an arc yx with label 1. These arcs provide the equality between the in-degree and the out-degree of each vertex in $\bar{H}$. Hence, the resulting digraph is Eulerian. Moreover, if G has a Hamiltonian circuit then we can remove the arcs of its associated circuit of label $0^{2|V(G)|}$ in $\bar{H}$ and the remaining graph is still Eulerian.

Therefore, G has a Hamiltonian circuit if and only if $\bar{H}$ has an Eulerian circuit starting at any vertex $r \in H$ with label smaller or equal to $0^{2|V(G)|}1^{|A(\bar{H})|-2|V(G)|}$.

3 A Linear Algorithm

In this section we assume that the arc-labeling gives a different label to each arc going out from the same vertex. We note that if we fix an initial vertex r, then there is a bijection between the trails starting at r and its labels.

We define the following greedy strategy to construct a circuit: Starting at a given vertex r, follow the unvisited arc (if exists) of minimal label. This strategy finishes with a trail, and this trail exhausts the vertex r. A trail constructed by this strategy is called an *alphabetic trail* starting at r.

Let U be a subset of vertices in G. A *cut* defined by U is the set of arcs with one end in U and the other in $V(G)\setminus U$, and is denoted by $\delta_G(U)$. For simplicity, for a trail T we write $\delta_G(T)$ instead of $\delta_G(V(T))$, where $V(T)$ is the set of the tail and head vertices of the arcs in T.

A vertex v is *exhausted* by a trail T if $\delta_{G\setminus A(T)}(v) = \emptyset$. We note that an alphabetic trail starting at r is the trail of lexicographically minimal label among all trails starting at r and exhausting r. We denote by $LastNotEx(T)$ the last vertex visited by T among all vertices not exhausted by T.

Let $T = e_1 \dots e_M$ be a trail and let e_i be an arc in the trail T. We denote by Te_i the subtrail $e_1 \dots e_i$, by e_iT the subtrail $e_i \dots e_M$ and by e_iTe_j the subtrail $e_i \dots e_j$ for $i < j$.

Lemma 2. *Let T be a circuit starting at r and exhausting r and let $v = LastNotEx(T)$. If e_i is the arc in T after the last visit to v then*

$$\delta_{G\setminus A(Te_{i-1})}(e_{i+1}T) = \{e_i\}$$

Proof. Let e be an arc of $\delta_G(e_{i+1}T)$. Since all vertices of $e_{i+1}T$ are exhausted by T, $e \in T$. Hence either $e \in Te_{i-1}$ or $e \in e_iT$. Therefore $e \in \delta_{G\setminus A(Te_{i-1})}(e_{i+1}T)$ if and only if $e = e_i$.

We note that these properties are valid for any trail starting at r and exhausting r, it does not need to be an Eulerian circuit.

For a trail $T = e_1 \dots e_M$ over G, we define a *failure* of T as a pair of arcs $e_i = vw, e_j = vx$ in the trail such that $i < j$ but $l(e_i) > l(e_j)$ and such that $\forall k < i$ with $e_k = vy$, $l(e_k) < l(e_j)$. The vertex v is called a *failure vertex*. Note that an alphabetic trail is a trail with no failures.

Our strategy to construct the Eulerian circuit starting at r of lexicographically minimal label is the following: Start at r with an alphabetic trail T_0. Let $v = LastNotEx(T_0)$ and let e_i be the arc in T after the last visit to v. Start at v an alphabetic trail T_1 over $G \setminus A(T_0)$. If the trail T_1 exhaust all its vertices, merge both trails obtaining $T_2 = (T_0 e_{i-1})T_1(e_i T_0)$ and repeat the process over $LastNotEx(T_2)$. If T_1 does not exhaust all its vertices, repeat the strategy recursively.

We note that if T_1 exhausts all its vertices, the trail $T_2 = (T_0 e_{i-1})T_1(e_i T_0)$ is the trail of minimal label exhausting r and having one failure.

This strategy can be stated as in Algorithm MinLex. Note that we include a global counter s and a bound $MaxSteps$ in order to count the number of failures of the resulting trail.

MinLex(A,r) : Compute the lexicographically minimal Eulerian circuit starting at r on $(V(G), A)$

Require: A an arc-subset of $A(G)$, r a vertex of G.
1. $s \leftarrow s + 1$
2. $T \leftarrow$ AlphabeticTrail(A, r)
3. **while** $NotEx(T) \neq \emptyset$ and $s \leq MaxSteps$
4. $\quad v \leftarrow LastNotEx(T)$
5. $\quad e_i \leftarrow$ the arc in T after the last visit to v
6. $\quad T \leftarrow (Te_{i-1})($MinLex$(A \setminus A(T), v))(e_i T)$
7. **end while**

Return: T

Where AlphabeticTrail(A, r) returns the alphabetic trail starting at r over $(V(G), A)$, $NotEx(T)$ is the set of not exhausted vertices in T, $MaxSteps$ is a fixed integer and s is a global counter initialized in 0.

In order to prove the correctness of our algorithm, we define O^k as the circuit of minimal label starting at r, having k failures and exhausting r and its failures vertices. In the following, we will define the trail O^{k-1} in terms of O^k.

First, we study the position of failures over O^k. The following lemma can be proved:

Lemma 3. *Let $\langle e_i, e_j \rangle$ and $\langle e_{i'}, e_{j'} \rangle$ be two different failures of O^k. Then it is not possible that $e_{i'} \in e_i O^k e_j$ and $e_{j'} \in e_j O^k$.*

The previous lemma state that two failures of O^k are either nested or in different subtrails of O^k. A failure $\langle e_i, e_j \rangle$ will be called simplicial if and only if there is not another failure in $e_i O^k e_j$. Therefore, there exist simplicial failures of O^k.

Let $\langle e_i, e_j \rangle$ be the first simplicial failure of O^k. We define $\tilde{O} = (O^k e_{i-1})(e_j O^k)$. We will prove that $\tilde{O} = O^{k-1}$. In order to prove this equality, we will prove some intermediate lemmas. All the proofs have the same idea: if the statement of the lemma is not fulfilled, then we can construct a trail with k failures with a label smaller than the label of O^k.

Lemma 4. *Let $\langle e_i, e_j\rangle$ be the first simplicial failure of O^k and let v be the vertex $LastNotEx(O^k)$. Then v is last visited in $O^k e_j$.*

Proof. (sketch) By contradiction, suppose that v is last visited by an arc $e_m \in e_{j+1}O^k$ and let W be the alphabetic trail starting at v over $G \setminus A(\tilde{O})$. Hence, the trail $(\tilde{O}e_{m-1})(W)(e_m\tilde{O})$ is a trail with k failures and a label smaller than the label of O^k, which is a contradiction.

Corollary 5. *Let $\langle e_i, e_j\rangle$ be the first simplicial failure of O^k and let v be the vertex $LastNotEx(\tilde{O})$. Then v is the failure vertex of $\langle e_i, e_j\rangle$.*

Proof. By contradiction, let x be a vertex in $e_j\tilde{O}$ not exhausted by $\tilde{O}$ and let e_n be the last visit of $\tilde{O}$ to x. By Lemma 4, the vertex x is exhausted by O^k, then there exists an arc $e_m \in e_iO^ke_j$ such that x is the head of e_{m-1} and the tail of e_m.

Therefore, the trail $(\tilde{O}e_{n-1})(e_mO^ke_{j-1})(e_iO^ke_{m-1})(e_n\tilde{O})$ uses the same arcs that O^k but its label is smaller than the label of O^k.

Lemma 6. *The trail $\tilde{O}$ is the trail O^{k-1}.*

Proof. (sketch) The trail $\tilde{O}$ starts and exhausts the vertex r and it has $k-1$ failures. By Corollary 5, $\tilde{O}$ exhausts all its failure. We only need to compare the label of $\tilde{O}$ with the label of O^{k-1}.

If the label of O^{k-1} is smaller than the label of $\tilde{O}$ then the arc producing this difference needs to be in a failure of $\tilde{O}$. Its corresponding failure vertex cannot occur before v, then $\tilde{O}e_{i-1} = O^{k-1}e_{i-1} = O^ke_{i-1}$.

Let $x = LastNotEx(O^{k-1})$. If x is last visited after e_j then we can merge an alphabetic trail starting at x over $G \setminus A(O^{k-1})$ and we obtain a trail with k failures and with a label smaller than the label of O^k, therefore $LastNotEx(O^{k-1}) = v$.

By Lemma 2, there is no arc between the vertices of subtrails $e_iO^ke_{j-1}$ and e_jO^{k-1}. Hence, if the label of e_jO^{k-1} is smaller than the label of $e_j\tilde{O}$, the trail $(O^{k-1}e_{i-1})(e_iO^ke_{j-1})(e_jO^{k-1})$ has k failures and a label smaller than the label of O^k. Therefore, $\tilde{O} = O^{k-1}$.

Now we are ready to prove the correctness of our algorithm.

Theorem 7. *The trail O^k is the trail obtained by the* MinLex*$(A(G), r)$ algorithm after k steps ($MaxSteps = k$).*

Proof. We prove this result by induction on the number of steps.

For $k = 0$, the algorithm produces the alphabetic trail starting at r, which is the minimal trail exhausting r with no failures.

Now we assume that $k > 0$. By Lemma 6, O^k is equal to merge O^{k-1} with the alphabetic trail over $G \setminus A(O^{k-1})$ starting at v, where $v = LastNotEx(O^{k-1})$. By induction hypothesis, O^{k-1} is obtained by the algorithm after $k-1$ steps. Moreover, the next step in the algorithm is an alphabetic trail starting at v over $G \setminus A(O^{k-1})$. Therefore, the resulting trail of the algorithm after k steps is exactly O^k.

Corollary 8. *The algorithm* MinLex$(A(G), r)$ *with a sufficient large integer* $MaxSteps$ *finishes with the Eulerian circuit of minimal label starting at* r

The complexity of our algorithm is linear in $|A(G)|$. We can use an adjacency list (see [12]) to represent the digraph, where each vertex v has a list with the head of each arc starting at v in the alphabetical order of its labels. Knowing this structure of a digraph, the algorithm can easily construct the alphabetic trails over $G \setminus O^i$ for every i, removing the visited arcs from the list and keeping track of exhausted vertices. Since this algorithm visits each arc at most twice, it can be implemented in $\mathcal{O}(|A(G)|)$, which is best possible.

4 An Application: Minimal De Bruijn Sequence

Given a set $\mathcal{D}$ of words of length n, a De Bruijn sequence of span n is a periodic sequence B such that every word in $\mathcal{D}$ (and no other n-tuple) appears exactly once in B. Historically, De Bruijn sequence was studied in an arbitrary alphabet considering the language of all the n-tuples. In [13] the concept of De Bruijn sequences was generalized to restricted languages with a finite set of forbidden substrings and it was proved the existence of these sequences and presented an algorithm to generate one of them. Nevertheless, it remained to find the minimal De Bruijn sequence in this general case.

In [14] was studied some particular cases where an alphabetic trail obtains the minimal De Bruijn sequence. Using Algorithm MinLex we can solve this problem efficiently in all cases.

A word p is said to be a *factor* of a word w if there exist words $u, v \in N^*$ such that $w = upv$. If u is the empty word (denoted by ε), then p is called a *prefix* of w, and if v is empty then is called a *suffix* of w.

Let $\mathcal{D}$ be a set of words of length $n+1$. We call this set a *dictionary*. A *De Bruijn sequence of span* $n+1$ for $\mathcal{D}$ is a (circular) word $B^{\mathcal{D},n+1}$ of length $|\mathcal{D}|$ such that all the words in $\mathcal{D}$ are factors of $B^{\mathcal{D},n+1}$. In other words,

$$\{(B^{\mathcal{D},n+1})_i \dots (B^{\mathcal{D},n+1})_{i+n \bmod (|\mathcal{D}|)} | i = 0 \dots |\mathcal{D}| - 1\} = \mathcal{D}$$

De Bruijn sequences are closely related to De Bruijn digraphs. The *De Bruijn graph of span* n, denoted by $G^{\mathcal{D},n}$, is the digraph with vertex set

$$V(G^{\mathcal{D},n}) = \{u \in N^n | u \text{ is a prefix or a suffix of a word in } \mathcal{D}\}$$

and arc set

$$A(G^{\mathcal{D},n}) = \{(\alpha v, v\beta) | \alpha, \beta \in N, \alpha v \beta \in \mathcal{D}\}$$

Note that the original definitions of De Bruijn sequences and De Bruijn graph given in [7] are the particular case of $\mathcal{D} = N^{n+1}$.

We label the arcs of the digraph $G^{\mathcal{D},n}$ using the following function l: if $e = (\alpha u, u\beta)$ then $l(e) = \beta$. This labeling has an interesting property: Let $T = v_0 e_0 \dots e_m v_{m+1}$ be a trail over $G^{\mathcal{D},n}$ of length $m \geq n$. Then T finishes in a vertex u if and only if u is a suffix of $l(T) = l(e_0) \dots l(e_m)$. This property

explains the relation between De Bruijn graphs and De Bruijn sequence: $B^{\mathcal{D},n+1}$ is the label of an Eulerian circuit of $G^{\mathcal{D},n}$. Therefore, given a dictionary $\mathcal{D}$, the existence of a De Bruijn sequence of span $n+1$ is characterized by the existence of an Eulerian circuit on $G^{\mathcal{D},n}$.

Let D be a dictionary such that $G^{\mathcal{D},n}$ is an Eulerian digraph. Let z be the vertex of minimum label among all vertices. Clearly, the minimal De Bruijn sequence has z as prefix. Hence, the minimal Eulerian circuit on $G^{\mathcal{D},n}$ starts at an (unknown) vertex and after n steps it arrives to z. Therefore if we start our Algorithm MinLex in the vertex z we obtain the Eulerian circuit of minimal label starting at z which have label $B = B' \cdot z$. Hence $z \cdot B'$ is the minimal De Bruijn sequence of span $n+1$ for $\mathcal{D}$.

References

1. Tutte, W.T.: Graph theory. Volume 21 of Encyclopedia of Mathematics and its Applications. Addison-Wesley Publishing Company Advanced Book Program, Reading, MA (1984)
2. Cheng, W.C., Pedram, M.: Power-optimal encoding fod DRAM address bus. In: ISLPED, ACM (2000) 250–252
3. Kari, J.: Synchronizing finite automata on Eulerian digraphs. Theoret. Comput. Sci. **295**(1-3) (2003) 223–232 Mathematical foundations of computer science (Mariánské Lázně, 2001).
4. Blazewicz, J., Hertz, A., Kobler, D., de Werra, D.: On some properties of DNA graphs. Discrete Appl. Math. **98**(1-2) (1999) 1–19
5. Pevzner, P.A.: L-tuple DNA sequencing: computer analysis. J. Biomol. Struct. Dyn. **7** (1989) 63–73
6. Pevzner, P.A., Tang, H., Waterman, M.S.: An eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences **98**(17) (2001) 9748–9753
7. de Bruijn, N.G.: A combinatorial problem. Nederl. Akad. Wetensch., Proc. **49** (1946) 758–764
8. Stein, S.K.: The mathematician as an explorer. Sci. Amer. **204**(5) (1961) 148–158
9. Bermond, J.C., Dawes, R.W., Ergincan, F.Ö.: De Bruijn and Kautz bus networks. Networks **30**(3) (1997) 205–218
10. Chung, F., Diaconis, P., Graham, R.: Universal cycles for combinatorial structures. Discrete Math. **110**(1-3) (1992) 43–59
11. Garey, M.R., Johnson, D.S.: Computers and intractability. W. H. Freeman and Co., San Francisco, Calif. (1979) A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.
12. Gibbons, A.: Algorithmic graph theory. Cambridge University Press, Cambridge (1985)
13. Moreno, E.: De Bruijn sequences and de Bruijn graphs for a general language. Inf. Process. Lett. **96** (2005) 214–219
14. Moreno, E., Matamala, M.: Minimal de Bruijn sequence in a language with forbidden subtrings. In Hromkovic, J., Nagl, M., Westfechtel, B., eds.: Graph-Theoretic Concepts in Computer Science. Volume 3353 of Lect. Notes in Comp. Sci., Springer-Verlag Heidelberg (2004) 168–176