Eulerian Paths with Regular Constraints

Orna Kupferman and Gal Vardi

School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel [*]

Abstract

Labeled graphs, in which edges are labeled by letters from some alphabet Σ, are extensively used to model many types of relations associated with actions, costs, owners, or other properties. Each path in a labeled graph induces a word in Σ^* – the one obtained by concatenating the letters along the edges in the path. Classical graph-theory problems give rise to new problems that take these words into account. We introduce and study the *constrained Eulerian path* problem. The input to the problem is a Σ-labeled graph G and a specification $L \subseteq \Sigma^*$. The goal is to find an Eulerian path in G that satisfies L. We consider several classes of the problem, defined by the classes of G and L. We focus on the case L is regular and show that while the problem is in general NP-complete, even for very simple graphs and specifications, there are classes that can be solved efficiently. Our results extend work on Eulerian paths with edge-order constraints. We also study the *constrained Chinese postman* problem, where edges have costs and the goal is to find a cheapest path that contains each edge at least once and satisfies the specification. Finally, we define and study the *Eulerian language* of a graph, namely the set of words along its Eulerian paths.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases Eulerian Paths, Regular Languages

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.63

1 Introduction

Many practical problems can be reduced to problems about graphs. A graph consists of vertices, which model objects, and edges, which model pairwise relations between the objects. Different settings call for different types of graphs. For example, when the relation between the objects is not symmetric, the graph is *directed*, and when it is not binary, the graph may have parallel edges or be *weighted*, say for modeling lengths or costs. In many applications, the edges of the graph carry information beyond weight. For example, edges may be associated with an action (say, in VLSI design), a query (say, in databases), properties like their owner or their security level (say, in a network of channels), and many more. Such applications require *labeled graphs*, in which each edge is labeled by a letter from some alphabet.[1]

Each path in a Σ-labeled graph induces a word in Σ^* – the one obtained by concatenating the letters along the edges in the path. Classical graph-theory problems give rise to new problems that take these words into account. For example, rather than finding any *shortest path* between two given vertices in a graph [10], it is sometimes desirable, say in transportation planning [5], web searching [1], or network routing, to restrict attention to paths that satisfy some constraint [4]. The basic query mechanism in these applications retrieves all pairs of nodes connected by a path conforming to a given pattern. There have been plenty of theoretical and practical work on the subject of *regular path queries*, where we wish to find all objects reachable by paths whose labels form a word in a

[*] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 278410, and from The Israel Science Foundation (grant no 1229/10).

[1] Alternatively, one could consider graphs with labels on vertices. It is not hard to alter our results to apply also for this setting.

given regular language over the alphabet of the labels [8]. As another example, rather than finding a *maximal flow* along arbitrary routes in a graph [13], one may want to restrict the used routes to ones that satisfy some specification [24]. The specification may restrict the length of the routes, preventing long routes from consuming the system, it may restrict the number of different resources applied in a route, require an application of specific resources, require a specific event to trigger another specific event, and so on. As a third example, an extension of network formation games [2] assumes that the edges in the network are labeled and allows to lift the reachability objectives of the players to objectives that are arbitrary regular languages [3]. Paths constrained by regular languages were also considered in the context of finding efficient algorithms for processing database queries ([25, 1]). Finally, online algorithms for finding paths that satisfy regular constraints are given in [7].

An interesting question is how enriching the setting with labels and constrains influences the complexity of classical problems. Note that now there are two parameters to the problem: the graph itself, as well as a formal language $L \subseteq \Sigma^*$, which is usually regular and is given by means of a finite automaton or a regular expression. We refer to L as a *specification*. Existing work shows that the picture is diverse. For example, in the case of shortest paths, finding a shortest path that satisfies regular and even context-free specifications can still be done in polynomial time [4]. Their algorithm is generalized in [6] for graphs with both negative and positive edge weights (but without negative-weight cycles). However, research in regular path queries shows that the problem of finding a shortest simple path that satisfies a regular specification is NP-complete (note that a shortest path that satisfies a specification need not be simple, even when all weights are positive) [25]. In the context of maximal flow, it is shown in [24] that even simple regular restrictions on the routes make the problem APX-hard, namely it is even hard to approximate. Likewise, adding regular objectives to network formation games result in games that are much less stable: they need not have a Nash Equilibrium, and their Price of Stability is higher than in the case of reachability objective [3].

An *Eulerian path* in a graph is a path that traverses all the edges of the graph, each edge exactly once. The problem of deciding whether a given graph has an Eulerian path (the EP problem, for short) was introduced in 1736, in what is considered the first paper in the history of graph theory. The problem can be solved in linear time by examining the parity of the degree of the vertices. Back in 1736, the motivation to study the problem was the challenge of traversing the seven bridges of Königsberg. Nowadays, the problem and its many variants have applications in planning [21], coding [9], synchronization [19], DNA sequencing [28], and many more. In many of these applications, it is useful to restrict attention to Eulerian paths that satisfy some constraint. For example, [22] studies Eulerian paths that satisfy precedence constraints on the edges, specified by linear orders on subsets of the edges. In [28], the input to the EP problem contains a set of paths, and the goal is to find an Eulerian path that contains all the paths in the set as sub-paths. Another related problem is studied in [17]: each edge e in a graph with m edges is associated with an interval $I_e = [l_e, h_e]$ inside $[1, m]$, and the goal is to find an Eulerian path so that the position of every edge e in the path belongs to I_e.

Once we move to consider labeled graphs, the type of restrictions can be much richer. Eulerian paths in labeled graphs were considered in [26], where the problem of finding an Eulerian cycle with a lexicographically minimal label is shown to be NP-complete. Note that the constraint in [26] is not given by means of a specification $L \subseteq \Sigma^*$. Rather, the letters in Σ are ordered and the constraint refers to the lexicographic order between the Eulerian cycles, possibly with respect to a given word. In this work we study Eulerian paths with regular constraints. Formally, the *constrained Eulerian path* problem (CEP problem, for short) is defined as follows: given a Σ-labeled graph G and a regular language $L \subseteq \Sigma^*$, find an Eulerian path in G that satisfies L. We consider several classes of the problem, according to the classes of G and L. We first show that the general CEP problem is NP-complete, and that it is NP-hard already for very restricted graphs and specifications: when the graph does not have parallel edges or loops, and when the specification is a regular expression of a fixed size that can be expressed by a two-state deterministic automaton. In fact, the problem stays NP-hard even when the specification L is a singleton (that is, requiring the Eulerian path to be labeled with a specific given word). We then describe classes of regular languages for which the CEP problem can be solved in polynomial time. For this, we relate the CEP problem with the problem of finding edge-disjoint paths in a graph. In particular, we show that the CEP problem can be solved in polynomial time for regular expressions of the form $R = b_1 \ldots b_k$, where k is fixed, for every $1 \leq i \leq k$, we have $b_i = \sigma_i^*$ or $b_i = \sigma_i$ for some $\sigma_i \in \Sigma$, and for every $\sigma \in \Sigma$, the expression σ^* appears at most three times in R. Alternatively, $b_i = w_i^*$ or $b_i = w_i$ for some $w_i \in \Sigma^*$, and every $\sigma \in \Sigma$ appears at most once in R. We demonstrate the usefulness of such expressions in specifying desired behaviors of paths. We also consider multi-labeled graphs, where an edge may be labeled by several letters, and show that then, the problem is NP-hard even for specifications given by a regular expression of the form a^*b^*, which essentially partitions the path into two types of labels.

An optimization variant or the EP problem is the *Chinese postman problem*. There, each edge in the graph has a non-negative cost, and the goal is to find a *postman path* – one that contains each edge in the graph at least once, of a minimal cost. Clearly, when the graph has an Eulerian path, it induces an optimal postman path. Otherwise, the goal is to get as close as possible to an Eulerian path. Researchers have studied useful restrictions on the allowed postman paths [12]. In particular, a natural extension of the precedence restrictions studied for the EP problem is the *hierarchical Chinese postman problem* [11, 16, 23]. Here, the edges of the graph are partitioned into clusters $E_1, \ldots, E_k$, and a precedence relation $\prec$ specifies the order in which the clusters should be traversed. That is, we seek the cheapest path that visits each edge at least once and so that if $E_i \prec E_j$, then all the edges in E_i are visited for the first time before an edge in E_j is visited. The problem is NP-hard in general, but can be solved in polynomial time in some cases. We consider labeled graphs and study the *constrained Chinese postman* problem (the CCP problem, for short), where the postman path should satisfy a regular specification. We study the complexity of the CCP problem, show that it is in general NP-complete, but point to useful polynomial cases.

A labeled graph G can be viewed as a generator of formal languages. The traditional way to do this is to designate some of the vertices of G as initial and as final vertices. The language of the obtained *automaton* is then the set of words that label paths from some initial to some final vertex. We introduce and study the *Eulerian language* of G, denoted $EL(G)$, namely the set of words that label Eulerian paths in G. Clearly, deciding whether $EL(G) \neq \emptyset$ amount to deciding whether G has an Eulerian path, and can be done in linear time. More interesting questions about the Eulerian language of G relate it to nontrivial languages: whether $EL(G)$ is contained in some specification L, whether some set L of desired behaviors is contained in $EL(G)$, and the relation between the Eulerian languages of two different graphs. Since $EL(G)$ contains only words of a fixed length, we know that $EL(G)$ is finite and hence regular. On the other hand, given a regular language $L \subseteq \Sigma^*$ it is not clear whether there is a graph G such that $EL(G) = L$. We study all the above problems and show that they belong to different levels of the polynomial hierarchy.

2 Preliminaries

2.1 Graphs and Eulerian paths

A *graph* $G = \langle V, E\rangle$ consists of a set V of vertices and a set E of directed edges. The graph G may contain loops and parallel edges.[2] A graph is *simple* if it does not contain loops or parallel edges. A *path* P in G is a sequence of edges $e_1, \ldots, e_k$ such that there are $k + 1$ vertices $v_0, \ldots, v_k$

[2] Since G may contain parallel edges, we do not refer to E as a subset of $V \times V$. However, for simplicity of notations, whenever there is no cause for confusion, we still denote an edge by a pair $(v_1, v_2) \in V \times V$.

and $e_i = (v_{i-1}, v_i)$ for all $1 \leq i \leq k$. We say that P is a path of length k from v_0 to v_k. If $v_0 = v_k$, then P is a *cycle*. We sometimes refer to P as a sequence of vertices, referring to the vertices $v_0, \ldots, v_k$. For an alphabet Σ, a Σ*-labeled graph* is a tuple $G = \langle V, E, l \rangle$, where $\langle V, E \rangle$ is a graph and $l : E \to \Sigma$ maps each edge to a letter in Σ. The label of a path $P = e_1, \ldots, e_k$, denoted $l(P)$, is the word $l(e_1) \cdot \ldots \cdot l(e_k)$ obtained by concatenating the labels of the edges along P. A *specification* for G is a language $L \subseteq \Sigma^*$. We say that P *satisfies* a specification L if $l(P) \in L$.

Consider a graph $G = \langle V, E \rangle$. For a vertex $v \in V$, the *in degree* and *out degree* of v, denoted $indeg(v)$ and $outdeg(v)$, are the number of edges entering and leaving v, respectively. An edge $(u_1, u_2) \in E$ is *incident to* v if $u_1 = v$ or $u_2 = v$. We say that G is *strongly connected* if for every two vertices $u, v \in V$ there is a path from u to v. An *undirected path* in G is a sequence of edges $e_1, \ldots, e_k$ such that there are $k + 1$ vertices $v_0, \ldots, v_k$ and $e_i \in \{(v_{i-1}, v_i), (v_i, v_{i-1})\}$ for all $1 \leq i \leq k$. We say that G is *connected* if for every $u, v \in V$ there is an undirected path from u to v. The *size* of G, denoted $|G|$, is the number of vertices and edges in G.

A path in a graph $G = \langle V, E \rangle$ is *Eulerian* (EP, for short) if it visits every edge in E exactly once. We say that G is *Eulerian* if it has an Eulerian cycle. For two edges e_1 and e_2 in E, an EP with $e_1 \prec e_2$ is an EP in which the edge e_1 is visited before the edge e_2. The following is well known.

▶ **Theorem 1.** *Consider a directed graph $G = \langle V, E \rangle$.*

1. *The graph G is Eulerian iff G is connected and for every $v \in V$, we have $indeg(v) = outdeg(v)$.*
2. *The graph G has an Eulerian path from s to t (with $s \neq t$) iff G is connected, $outdeg(s) = indeg(s)+1$, $indeg(t) = outdeg(t)+1$, and for every $v \notin \{s, t\}$, we have $indeg(v) = outdeg(v)$.*

By Theorem 1, one can decide in time linear in $|G|$ whether G is Eulerian or has an EP between two given vertices. Furthermore, if G has an EP from s to t, then such path can be found as follows: Follow some path from s until reaching t. It is not possible to get stuck at any vertex other than t, because when the path enters another vertex v there must be an unused edge leaving v. The path formed in this way may not cover all edges of the initial graph. As long as there exists a vertex v that belongs to the current path but that has incident edges not part of the path, start another path from v, following unused edges until returning to v (as above, this process does not get stuck), and join the path formed in this way to the previous path.

A path (cycle) in a graph $G = \langle V, E \rangle$ is *Hamiltonian* if it visits every vertex in V exactly once. The Hamiltonian path (cycle) problem, namely deciding whether a given graph has a Hamiltonian path (cycle), is NP-hard already when the graph is simple [15].

2.2 Regular languages and the constrained Eulerian path problem

A *nondeterministic finite automaton* (NFA, for short) is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, F \rangle$, where Σ is a finite alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, and $F \subseteq Q$ is a set of final states. Given a word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_l \in \Sigma^*$, a *run* of $\mathcal{A}$ on w is a sequence $r = q_0, q_1, \ldots, q_l$ of states such that $q_0 \in Q_0$ and $q_{i+1} \in \delta(q_i, \sigma_{i+1})$ for all $i \geq 0$. The run is accepting if $q_l \in F$. The NFA $\mathcal{A}$ *accepts* the word w iff it has an accepting run on it. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$ is the set of words that $\mathcal{A}$ accepts. If $|Q_0| = 1$ and $|\delta(q, \sigma)| \leq 1$ for all $q \in Q$ and $\sigma \in \Sigma$, then $\mathcal{A}$ is *deterministic*. Note that a deterministic finite automaton (DFA) has at most one run on each word.

A *regular expression* (RE, for short) over Σ is defined as follows.

- $\emptyset$, ϵ, and σ, for $\sigma \in \Sigma$, are REs.
- If R_1 and R_2 are REs, then so are $R_1 + R_2$, $R_1 \cdot R_2$, and R_1^*.

The language of an RE R, denoted $L(R)$, is defined inductively on the structure of R, where $+$, $\cdot$, and $*$ stand for union, concatenation, and Kleene star, respectively.

We say that R is a *chain* RE if it is of the form $b_1 \cdot b_2 \cdots b_k$, where for every $1 \leq i \leq k$, we have $b_i = w_i^*$ or $b_i = w_i$ for a word $w_i \in \Sigma^*$. We call every such b_i a *block*. Note that a chain RE does not contain the symbol "$+$" and does not contain nested Kleene stars. For $l \geq 1$, we say that a chain RE R is l-*wide* if $|w_i| \leq l$ for all $1 \leq i \leq k$ with $b_i = w_i^*$. Note that when R is 1-wide, then for all $1 \leq i \leq k$, we have that $b_i = \sigma^*$ or $b_i = \sigma$, for some $\sigma \in \Sigma$. For $d \geq 1$, we say that R is d-*diverse* if each letter in Σ appears in R at most d times. We say that R is d-*star-diverse* if for each letter $\sigma \in \Sigma$, the expression σ^* appears in R at most d times. Note that if R is d-diverse then it is also d-star-diverse. For example, $a^*(ba)^*c^*(bc)^*$ is a 2-wide, 2-diverse, 1-star-diverse chain RE. Then, $a^*b^*a^*$ is a 1-wide, 2-diverse, 2-star-diverse chain RE, and $(a+b)^*(b+c)^*$ is not a chain RE.

The *constrained Eulerian path* problem (CEP problem, for short) is defined as follows: given a labeled graph G and a regular language L, given by means of an NFA, DFA, or RE, find an Eulerian path in G that satisfies L.

▶ **Example 2.** We describe some chain REs that are useful specifications.
[Zone patrolling and periodic checks] Consider a communication, social, or physical network. Let a, b, c, and d be labels of edges in different zones of the network. We may want to patrol the network in a certain pattern. For example, in security, one may want a guard to patrol the zones of a physical network in a certain order, and in commercial applications, one may want to do the same with an advertisement that traverses a social network. Likewise, several communication protocols are based on the fact that a message must patrol different zones of the network in some predefined order before reaching its destination; e.g., in Onion routing, where the message is encrypted in layers, or in *proof-of-work* protocols that are used to deter denial of service attacks and other service abuses such as spam. For this, REs of the form $(a^+b^+c^+d^+)^*$, where σ^+ stands for $\sigma\sigma^*$, may be useful. If the pattern of visits is of a known bounded length, it can be specified as a conjunction of 1-wide chain REs, say $a^+b^+c^+d^+a^+b^+c^+d^+$.

As another example, let s label edges in which a checksum is performed on the message, and let σ label all other edges. We may want a message to be periodically checked for corruptions. This can be specified by the chain RE $(\sigma^i s)^*$, for the duration i after which a check should be performed. If we want the specification to be more flexible, say allow different durations between successive checks, all bounded by i, this is possible, but the RE is no longer a chain RE.
[Bounded-delay response and FIFO] Let r, r_1, and r_2 label edges in which requests are submitted, possibly parameterized by the user that submits them, and let g, g_1, and g_2 label edges in which requests are granted, again possibly parameterized by the granted user. The semantics of requests and grants depend on the type of the network. Suppose there can be at most one request in the graph and we want a request, if submitted, to be followed by a grant within 3 steps. Let σ label all edges that are not labeled r or g. This can be specified by $R = \sigma^* + \sigma^* r g \sigma^* + \sigma^* r \sigma g \sigma^* + \sigma^* r \sigma \sigma g \sigma^*$. Note that R is a union of 1-wide 4-diverse chain REs. In case of a grant bounded by k steps, the REs are 1-wide $(k+1)$-diverse chain REs. If there are two requests and we want them to be granted in a FIFO order, the specification is $R = \sigma^* + \sigma^* r_1 \sigma^* g_1 \sigma^* + \sigma^* r_1 \sigma^* r_2 \sigma^* g_1 \sigma^* g_2 \sigma^* + \sigma^* r_1 \sigma^* g_1 \sigma^* r_2 \sigma^* g_2 \sigma^*$, and dually when only r_2 is submitted or when r_2 is before r_1. Note that R is a union of 1-wide 5-diverse chain REs.

3 It Is Hard

In this section we study the general CEP problem and show that it is NP-complete for specifications given by NFAs, DFAs, or REs. The reductions in this section are simple. We give them here for completeness and as a warm-up before things get more complicated in the next sections.

▶ **Theorem 3.** *The CEP problem is NP-complete.*

Proof. We start with membership in NP. Checking the membership of a given word in the language of a given NFA, DFA, or RE can be done in polynomial time. Hence, given a labeled graph $G = \langle V, E, l \rangle$ and a regular language L given by means of an NFA, DFA, or RE, checking whether a sequence P of $|E|$ edges is an EP in G and that $l(P) \in L$ can be done in polynomial time.

We prove NP-hardness by a reduction from the Hamiltonian-path problem. We prove hardness for specifications given by DFAs. Hardness for NFAs follows immediately, and hardness for REs follows from the polynomial translation of DFAs to REs. Given a graph $G = \langle V, E \rangle$, we construct a labeled graph G' and a DFA $\mathcal{A}$ such that there is a Hamiltonian path in G iff there is an EP in G' that satisfies $L(\mathcal{A})$. The labeled graph G' is over the alphabet V. It consists of a single vertex u with $|V|$ parallel self loops, each labeled by a different vertex in V. That is, the EPs of G' correspond to permutations of V. We define the specification DFA $\mathcal{A}$ so that $L(\mathcal{A})$ includes exactly all words that label paths in G. It is easy to define $\mathcal{A}$ as above by adding to G an initial state that has a transition to all vertices, and labeling all transitions, including the new ones, by their destination vertex. All the states in $\mathcal{A}$ are final. Now, there is a Hamiltonian path in G iff there is a permutation of V that forms a path in G iff there is an EP in G' that satisfies $L(\mathcal{A})$. ◀

The graph G' used in the proof of Theorem 3 is not simple. It is easy, however, to add a new letter $\#$ to the alphabet V and obtain a simple graph G'' by replacing every (parallel) self-loop labeled v in G' by a (disjoint) cycle labeled $v\#$ in G''. Hence the following theorem (see Appendix A.1 for the detailed proof).

▶ **Theorem 4.** *The CEP problem is NP-hard already for simple graphs.*

4 It Is Very Hard

While the graph G'' defined in the reduction in the proof of Theorem 4 is simple, the DFA $\mathcal{A}$ used in the proof is essentially the graph G. This suggests that we may do better with specifications of a constant size. In this section we show that the CEP problem is NP-hard already for more restricted cases, in particular for specification of a constant size. As we then show in Section 5, the cases we point to are tight, in the sense that tightening the restrictions makes the problem feasible.

We first define the *Eulerian closure* of a given graph – a construction that is going to be useful in some of our reductions. Given a graph $G = \langle V, E \rangle$, the *Eulerian closure of* G is the graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, defined as follows (see an Example in Figure 1).

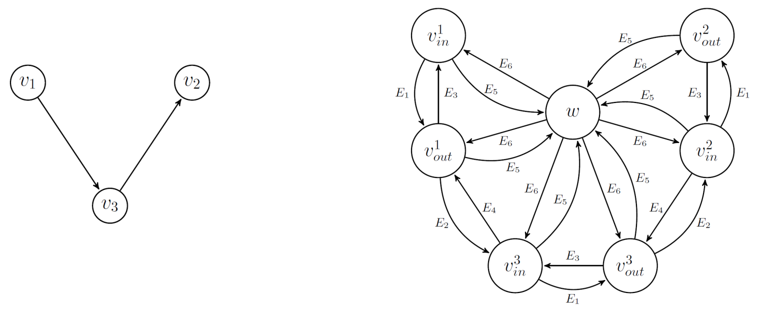

Figure 1 A graph and its Eulerian closure.

For each vertex $v \in V$ we include in $\mathcal{V}$ two vertices v_{in} and v_{out}. We also include in $\mathcal{V}$ a new vertex w. That is, $\mathcal{V} = \{v_{in} : v \in V\} \cup \{v_{out} : v \in V\} \cup \{w\}$. The set of edges of $\mathcal{G}$ is the union $\mathcal{E} = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6$, defined below:

- $E_1 = \{(v_{in}, v_{out}) : v \in V\}$. Thus, E_1 includes an edge for each vertex in G.
- $E_2 = \{(u_{out}, v_{in}) : (u, v) \in E\}$. Thus, E_2 includes an edge for every edge in G.

- $E_3 = \{(v_{out}, v_{in}) : v \in V\}$. Thus, E_3 includes an opposite edge for every edge in E_1.
- $E_4 = \{(v_{in}, u_{out}) : (u, v) \in E\}$. Thus, E_4 includes an opposite edge for every edge in E_2.
- $E_5 = \{(v, w) : v \in \mathcal{V} \setminus \{w\}\}$.
- $E_6 = \{(w, v) : v \in \mathcal{V} \setminus \{w\}\}$.

Note that, by Theorem 1, the graph $\mathcal{G}$ is Eulerian. Indeed, the edges in E_5 and E_6 guarantee that $\mathcal{G}$ is strongly connected regardless of the connectivity of G. Also, for every vertex $v \in \mathcal{V}$, we have $indeg(v) = outdeg(v)$. Finally, if G is simple, then so is $\mathcal{G}$.

▶ **Theorem 5.** *The CEP problem is NP-hard already for a simple graph and a specification given by fixed-size RE that can be expressed by a DFA with two states.*

Proof. We show a reduction from the problem of Hamiltonian path for simple graphs. Given a simple graph $G = \langle V, E\rangle$, we construct a simple labeled graph G' and a RE R that can be expressed by a DFA with two states, such that there is a Hamiltonian path in G iff there is an EP in G' that satisfies $L(R)$. The graph G' is defined by $G' = \langle \mathcal{V}, \mathcal{E}, l\rangle$ where $\langle \mathcal{V}, \mathcal{E}\rangle$ is the Eulerian closure of G, and $l(e)$ is $\{a\}$ if $e \in E_1$, is $\{b\}$ if $e \in E_2$, and is $\{c\}$ otherwise. Let $R = (a + b)^*(b + c)^*$. Note that $L(R)$ can be expressed by a DFA with two states. In Appendix A.2, we prove that the reduction is correct. For this, we show that the way we have defined the Eulerian closure of G guarantees that if there is a Hamiltonian path in G, then it induces a path in G' that contains only edges labeled by a or b and can be extended to an Eulerian cycle by appending a path that contains only edges labeled by b or c. Also, every EP in G' that satisfies $L(R)$ starts with a subpath that induces a Hamiltonian path in G. ◀

We continue and show that the CEP problem is NP-hard already for singleton specifications, namely when the specification consists of a single word, and for specifications given by a fixed-size RE without union and without nested Kleene star operators.

▶ **Theorem 6.** *The CEP problem is NP-hard for singleton specifications and for specifications given by a fixed-size* 2*-wide* 2*-diverse chain RE.*

Proof. In both cases, we show a reduction from the problem of Hamiltonian path for simple graphs. Let $G = \langle V, E\rangle$ be a simple graph, and let $V = \{v^1, \ldots, v^n\}$. We define the simple graph $G' = \langle \mathcal{V}, \mathcal{E}, l\rangle$, where $\langle \mathcal{V}, \mathcal{E}\rangle$ is the Eulerian closure of G, and $l(e)$ is $\{a\}$ if $e \in E_1$, is $\{b\}$ if $e \in E_2 \cup E_6$, and is $\{c\}$ otherwise.

Consider the word $x = a(ba)^{n-1}c^{2n-1}(cb)^{|E|+n+1}$. We prove that there is a Hamiltonian path in G iff there is an EP in G' that is labeled by x. First, it is not hard to see that if G' has an EP labeled x, then the prefix $a(ba)^{n-1}$ of x induces a Hamiltonian path in G. Now, assume that there is a Hamiltonian path P in G from v^1 to v^n. We construct an Eulerian cycle in G' as follows: The Eulerian cycle starts with the path Q in G' corresponding to P. This path starts with the edge (v^1_{in}, v^1_{out}), ends with the edge (v^n_{in}, v^n_{out}), and visits the edge (v^i_{in}, v^i_{out}) for every i exactly once. Note that $l(Q) = a(ba)^{n-1}$. In Appendix A.3, we show how the way the Eulerian closure is defined enables us to continue Q as required.

We continue to the second class. Let R be the RE $R = a(ba)^*c^*(cb)^*$. Note that R is indeed a fixed-size 2-wide 2-diverse chain RE. We show that there is a Hamiltonian path in G iff there is an EP in G' that satisfies $L(R)$. Again, it is not hard to see that if G' has an EP that satisfies $L(R)$, then the prefix $a(ba)^*$ of R induces a Hamiltonian path in G. Also, if there is a Hamiltonian path in G, then an Eulerian cycle in G' that satisfies $L(R)$ can be constructed as in the case of the word x.[3] ◀

[3] Note that we could have defined the RE R to be $a^*(ba)^*c^*(cb)^*$, implying that NP-hardness applies already for fixed-size 2-wide 2-diverse chain REs in which all blocks have a Kleene star.

5 But Sometimes It Is Easy

In this section we show classes of regular languages for which the CEP problem can be solved in polynomial time. By Theorem 6, the CEP problem is NP-hard for fixed-size 2-wide 2-diverse chain REs. We show that when one of the width and diversity parameters is tightened, the problem becomes feasible. We start with diversity and show that when a chain RE R is 1-diverse, we can solve the CEP problem for it in polynomial time even when R and its blocks are not of a fixed size.

▶ **Theorem 7.** *The CEP problem can be solved in polynomial time for specifications given by a* 1*-diverse chain RE.*

Proof. Let $G = \langle V, E, l\rangle$ be a labeled graph and let $R = b_1 \dots b_k$ be a RE, where for every i we have $b_i = w_i^*$ or $b_i = w_i$ for some $w_i \in \Sigma^*$, and every $\sigma \in \Sigma$ appears at most once in R. We assume that every letter that appears in G, appears also in R, because otherwise the CEP problem is trivial. We show how to find an EP that satisfies $L(R)$ and that starts in a vertex $v_1 \in V$. Note that since every $\sigma \in \Sigma$ appears at most once in R, then the subpath that corresponds to a block b_i must be an EP in the subgraph G_i induced by the edges labeled by letters in w_i. Therefore, the first vertex in every subpath determines the last vertex in this subpath: if the degrees in every vertex in G_i are balanced, that is, the in degree is equal to the out degree for every vertex, then an EP must be a cycle; if the degrees are not balanced then an EP must start in the only vertex s_i where $outdeg(s_i) = indeg(s_i) + 1$ and end in the only vertex t_i where $indeg(t_i) = outdeg(t_i) + 1$. Thus, the algorithm checks whether G_1 has an EP from v_1 that corresponds to b_1, and if it does then it finds the vertex v_2 in which this EP ends. Then the algorithm checks whether G_2 has an EP from v_2 that corresponds to b_2 and continues similarly.

We now show how to find an EP in $G_i = \langle V_i, E_i, l\rangle$ that starts in a vertex v_i and corresponds to b_i. If $b_i = w_i$, then we check whether G_i contains every letter in w_i exactly once, and whether w_i induces a path in G_i from the vertex v_i. Assume now that $b_i = w_i^*$, and $w_i = \sigma_0 \dots \sigma_{n-1}$. We construct a graph $G'_i = \langle V'_i, E'_i\rangle$, where $V'_i = \{\langle v, j\rangle : v \in V_i \text{ and } 0 \leq j \leq n-1\}$ and $E'_i = \{(\langle u, j\rangle, \langle v, j+1 \ (mod\ n)\rangle) : (u, v) \in E_i \text{ and } l((u, v)) = \sigma_j\}$. Thus, G'_i includes n copies of G_i such that every edge (u, v) with $l((u, v)) = \sigma_j$ in G_i induces an edge from u in the j-th copy to v in the $(j+1)$-th copy in G'_i. Note that there is an EP in G_i from the vertex v_i that corresponds to b_i, iff there is an EP in G'_i from the vertex $\langle v_i, 0\rangle$ to some vertex $\langle u, 0\rangle$. Therefore, the problem is reduced to finding an EP from $\langle v_i, 0\rangle$ in G'_i. ◀

We continue and show that tightening the width also makes the problem solvable in polynomial time. We first describe another well-studied problem that we show to be strongly related to our problem. Let $G = \langle V, E\rangle$ be a directed or undirected graph and let (s_i, t_i), for $i = 1, \dots, k$, be k ordered pairs of vertices. In the *edge-disjoint paths* problem (EDP problem, for short), we need to find, for every $1 \leq i \leq k$, a path in G from s_i to t_i such that the paths are edge-disjoint; that is, an edge cannot appear in more than one path. The EDP problem is NP-complete for both directed and undirected graphs [30]. When the graph is undirected and k is fixed, there is a polynomial-time algorithm ([29, 20]). For directed graphs, the problem is NP-complete already when $k = 2$ [14]. The directed graph $G_D = \langle V, E_D\rangle$ where $E_D = \{(t_i, s_i) : i = 1 \dots k\}$ is called the *demand graph*.

Consider the graph $G + G_D = \langle V, E \cup E_D\rangle$ obtained by adding to G the edges from G_D. When $G + G_D$ is Eulerian, we say that there is an *Eulerian promise on the demand*. It is shown in [18] that when there is an Eulerian promise on the demand, there is a polynomial-time algorithm for the EDP problem with $k = 3$. It is also conjectured in [18] that when there is an Eulerian promise on the demand, there is a polynomial-time algorithm for the EDP problem for every fixed k. To the best of our knowledge, however, this problem is still open (it is also declared open in [30, 27]).

We first relate the EDP problem to the problem of finding an EP that respects a linear order on the subset of the edges. The proof of Lemma 8 can be found in Appendix A.4.

▶ **Lemma 8.** *Consider a directed graph $G = \langle V, E \rangle$ and two vertices $s, t \in V$. Let $e_1, \ldots, e_k$ be some edges in E and $\tau = e_1 \prec \ldots \prec e_k$ be an order on them. If $k \leq 2$, then finding an EP from s to t that respects τ can be done in polynomial time. If $k > 2$ is fixed, then finding an EP from s to t that respects τ can be solved in polynomial time iff the EDP problem for a directed graph with an Eulerian promise on the demand can be solved in polynomial time for $k + 1$ paths.*

We now relate the CEP problem for fixed-size 1-wide chain REs to the problem of finding an EP that respects a linear order on a subset of the edges. Lemma 8 then enables us to relate the former also to the EDP problem, implying that restricting the width also leads to a polynomial complexity.

▶ **Theorem 9.** *The CEP problem can be solved in polynomial time for specifications given by a fixed-size 1-wide 3-star-diverse chain RE. For specifications given by a fixed-size 1-wide d-star-diverse chain RE, the CEP problem can be solved in polynomial time iff the EDP problem for a directed graph with an Eulerian promise on the demand can be solved in polynomial time for d paths.*

Proof. Let $R = b_1 \ldots b_k$ for a fixed k, where for every i we have $b_i = \sigma_i^*$ or $b_i = \sigma_i$ for some $\sigma_i \in \Sigma$. We assume that every letter that appears in G appears also in R. Indeed, otherwise the CEP problem is trivial. We run over all the options for choosing $k + 1$ vertices $v_0, \ldots, v_k \in V$ (there are $|V|^{k+1}$ such options), and check whether G has an EP from v_0 to v_k that satisfies $L(R)$ and can be partitioned into k subpaths, such that the i-th subpath starts in v_{i-1}, ends in v_i and corresponds to b_i. We now describe how to perform this check.

For every i such that $b_i = \sigma$ for some $\sigma \in \Sigma$, the graph G must have an edge $e = (v_{i-1}, v_i)$ with $l(e) = \sigma$. All the other subpaths corresponding to b_j for $j \neq i$, cannot use the edge e. In particular, if b_j with $j \neq i$ is a single-letter block, then it cannot use the edge e. In the rest of this proof we assume that for every single-letter block $b_n = \sigma_n$ there is an edge $e_n = (v_{n-1}, v_n)$ such that $l(e_n) = \sigma_n$, and that if b_m is a single-letter block with $m \neq n$ then e_m and e_n are different edges (they can be parallel edges). We denote the set of edges that correspond to a single-letter block by E_s.

Let $\sigma \in \Sigma$ and let $b_{i_1}, \ldots, b_{i_m}$ be the blocks in R such that for every $1 \leq j \leq m$ we have $b_{i_j} = \sigma^*$. Let G_σ be the subgraph of G induced by the edges $\{e \in E \setminus E_s : l(e) = \sigma\}$. Let G'_σ be the graph obtained from G_σ by adding, for every $1 \leq j \leq m - 1$, an edge $e_{i_j} = (v_{i_j}, v_{i_{j+1}-1})$; that is, we add for every i_j an edge from the end of the block i_j to the beginning of the block i_{j+1}. We check whether there is an EP in G'_σ from v_{i_1-1} to v_{i_m} such that $e_{i_1} \prec e_{i_2} \prec \ldots \prec e_{i_{m-1}}$. In Appendix A.5, we prove that such an EP exists in G'_σ for every σ iff the required EP in G exists.

Thus, we reduce the CEP problem to the problem of finding an EP that respects a linear order on a subset of the edges. By Lemma 8, the latter can be reduced to the EDP problem. Accordingly, we have a polynomial-time algorithm if for every $\sigma \in \Sigma$ the expression σ^*, appears at most three times in R (that is, R is 3-star-diverse). Also, if the EDP problem for a directed graph with an Eulerian promise on the demand can be solved in polynomial time for d paths, then, according to Lemma 8, we have a polynomial-time algorithm also if R is d-star-diverse.

Now, assume that there is a polynomial-time algorithm for the case R is d-star-diverse. Let $H = \langle V_H, E_H \rangle$ be a graph and let $e_1, \ldots, e_{d-1} \in E_H$. Let H' be a labeled graph obtained from H by assigning a label σ_i for every edge e_i, and assigning a label σ for every other edge in E_H. Note that H has an EP with $e_1 \prec \ldots \prec e_{d-1}$ iff H' has an EP that satisfies $L(R)$ for $R = \sigma^*\sigma_1\sigma^*\sigma_2 \ldots \sigma^*\sigma_{d-1}\sigma^*$. By our assumption, the latter problem can be solved in polynomial time. Therefore, by Lemma 8, the EDP problem for a directed graph with an Eulerian promise on the demand can be solved in polynomial time for d paths. ◀

We conclude that there is a polynomial-time algorithm for every specification that is a disjunction of polynomially many REs of the forms handled in Theorems 7 and 9. As demonstrated in

Example 2, such disjuncts can specify useful behaviors. We note that with some additional ''technical acrobatics", it is possible to squeeze the lemon some more and point to additional classes of chain REs that can be solved in polynomial time. For example, if $R = b_1 \dots b_k$, where k is a fixed number and for every $1 \leq i \leq k$ we have $b_i = w_i^*$ for some $w_i \in \Sigma^*$, or $b_i = \sigma_i$ for some $\sigma_i \in \Sigma$, then it is possible to restrict the diversity of the letters, but allow repetitions of identical blocks, so that polynomial-time algorithms can be obtained by combining ideas used in the proofs of Theorems 9 and 7. We do not find, however, these special cases or technical acrobatics too interesting. A good intuitive and practical conclusion is that when the specification is a chain RE, it is recommended to use the ideas here in order to find the complexity of the CEP problem for it, and possibly to decompose or alter it to a disjunction of specifications of lower width and diversity for which a polynomial algorithm is possible.

6 Multi-Labeled Graphs

A *multi-labeled graph* is a tuple $G = \langle V, E, l\rangle$ where $\langle V, E\rangle$ is a graph and $l : E \to 2^\Sigma$ maps every edge to a subset of letters from the alphabet Σ. For a path $P = e_1, \dots, e_k$ we define $l(P) = \{\sigma_1 \dots \sigma_k : \sigma_i \in l(e_i) \text{ for every } 1 \leq i \leq k\}$. We say that P satisfies a specification $L \subseteq \Sigma^*$ if $l(P) \cap L \neq \emptyset$. Note that we take here the existential approach in model checking, where satisfaction amounts to an existence of a correct path.

We show that if the graph is multi-labeled, the CEP problem is NP-hard already for the RE a^*b^*. Note that a^*b^* is a fixed-size 1-wide 1-diverse chain RE. Thus, by both Theorems 9 and 7, the CEP problem in graphs in which each edge is labeled by a single letter can be solved in polynomial time.

▶ **Theorem 10.** *The CEP problem for multi-labeled simple graphs is NP-hard already for the specification given by the RE $R = a^*b^*$.*

Proof. We show a reduction from the problem of Hamiltonian path for simple graphs. Given a simple graph $G = \langle V, E\rangle$, we construct a multi-labeled simple graph G', such that there is a Hamiltonian path in G iff there is an EP in G' that satisfies $L(R)$. The graph G' is defined by $G' = \langle \mathcal{V}, \mathcal{E}, l\rangle$ where $\langle \mathcal{V}, \mathcal{E}\rangle$ is the Eulerian closure of G, and $l(e)$ is $\{a\}$ if $e \in E_1$, is $\{a, b\}$ if $e \in E_2$, and is $\{b\}$ otherwise. In Appendix A.6, we prove the correctness of the reduction. ◀

7 The Constrained Chinese Postman Problem

A *weighted graph* is a tuple $G = \langle V, E, c\rangle$, where $\langle V, E\rangle$ is a graph and $c : E \to \mathbb{R}^+$ maps every edge to a non-negative cost. The cost of a path $P = e_1, \dots, e_k$, denoted $c(P)$, is $\sum_{i=1}^{k} c(e_i)$; that is, the sum of the costs of the edges along the path. A *postman path* in G is a path that visits every edge in E at least once. An *optimal postman path* is a least-cost postman path. Similar definitions apply to cycles. In the well-studied *Chinese postman problem*, we are given a weighted graph G and need to find an optimal postman path. Clearly, when G has an EP, it induces an optimal postman path. Otherwise, the goal is to get as close as possible to an EP. Thus, the Chinese postman problem can be viewed as an optimization variant of the EP problem. The combinatorial simplicity of the EP problem is carried over to the Chinese postman problem. In particular, it has a well-known polynomial-time algorithm [12]. For completeness, we describe this algorithm in Appendix A.7.

A *labeled weighted graph* is a tuple $G = \langle V, E, l, c\rangle$, where $\langle V, E, l\rangle$ is a labeled graph and $\langle V, E, c\rangle$ is a weighted graph. For a regular language L, an *L-postman path* in G is a path that satisfies L and visits every edge in E at least once. An *optimal L-postman path* is a least-cost L-postman path. In the *constrained Chinese postman problem* (CCP problem, for short), we are given a labeled weighted graph G and a regular language L and need to find an optimal L-postman path in G. In this section we study the CCP problem.

First, we show that the corresponding decision problem is NP-complete, and is NP-hard already for restricted classes of graphs and specifications.

▶ **Theorem 11.** *Consider a labeled weighted graph G and a regular language L given by an NFA, DFA, or RE. For $k \in \mathbb{R}^+$, deciding whether there is an L-postman path P with $c(P) \leq k$ is NP-complete. Furthermore, it is NP-hard already when G is simple, when L is a singleton, when L is given by a DFA with two states, and when L is given by a fixed-size 2-wide 2-diverse chain RE.*

Proof. First, note that a labeled graph $H = \langle V_H, E_H, l_H \rangle$ has an EP that satisfies L iff the labeled weighted graph $H' = \langle V_H, E_H, l_H, c \rangle$ with $c(e) = 1$ for every $e \in E_H$ has an L-postman path with cost $|E|$. Thus, the lower bounds follow from Theorems 4, 5, and 6.

We prove the upper bound for L given by an NFA $\mathcal{A}$. The other cases follow. Let $G = \langle V, E, l, c \rangle$, $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, F \rangle$, and let $\mathcal{N}$ be the product NFA of $\mathcal{A}$ and G. Formally, $\mathcal{N} = \langle \Sigma, Q \times V, Q_0 \times V, \delta', F \times V \rangle$, where $\delta'(\langle q, v \rangle, \sigma) = \{\langle q', v' \rangle : q' \in \delta(q, \sigma), (v, v') \in E$, and $l((v, v')) = \sigma\}$. Note that $\mathcal{N}$ ignores parallel edges in G. Also, a path P in G satisfies $L(\mathcal{A})$ iff there is an accepting run in $\mathcal{N}$ whose projection on V corresponds to P. We claim that there is an $L(\mathcal{A})$-postman path P in G with $c(P) \leq k$ iff there is an $L(\mathcal{A})$-postman path $P' = e_1, \ldots, e_n$ in G with $c(P') \leq k$ of length $n \leq |E| \cdot |Q| \cdot |V|$. Let r be an accepting run in $\mathcal{N}$ whose projection on V corresponds to a path $P = e_1, \ldots, e_n$ in G, and assume that P includes every edge in G (including parallel edges) at least once. Let $i_1 < \ldots < i_{|E|}$ be the indices in which all edges appear for the first time in P. If for some j there are more than $|Q| \cdot |V|$ states between the appearance in r of the transition that corresponds to the edge e_{i_j} and the appearance of the transition that corresponds to the edge $e_{i_{j+1}}$, then r has a loop that can be avoided. By removing these loops we end up with a path of length $n \leq |E| \cdot |Q| \cdot |V|$. Thus, a witness for having an $L(\mathcal{A})$-postman path P in G with $c(P) \leq k$ is of size at most $n \leq |E| \cdot |Q| \cdot |V|$. ◀

Since the CCP problem is at least as hard as the CEP problem, we turn to consider cases for which the CEP problem is solvable in polynomial time. In particular, we restrict further the class of fixed-size 2-wide 2-diverse chain RE. First by restricting the width, and then the diversity.

▶ **Theorem 12.** *The CCP problem can be solved in polynomial time for specifications given by a fixed-size 1-wide 2-star-diverse chain RE.*

Proof. Let $G = \langle V, E, l, c \rangle$ and let $R = b_1 \ldots b_k$ be a 1-wide 2-star-diverse chain RE. We assume that every letter that appears in G, appears also in R. Indeed, otherwise the CCP problem is trivial. We run over all the (polynomially many) options for choosing $k + 1$ vertices $v_0, \ldots, v_k \in V$, and find a least-cost path in G from v_0 to v_k that satisfies $L(R)$, contains all edges, and can be partitioned into k subpaths such that the i-th subpath starts in v_{i-1}, ends in v_i, and satisfies $L(b_i)$.

First, assume that $b_i = \sigma_i^*$ for every i; that is, R does not contain single-letter blocks. Let $\sigma \in \Sigma$. If σ^* appears exactly once in R and $\sigma = \sigma_i$, then we find an optimal postman path from v_{i-1} to v_i in the subgraph G_σ induced by the edges in G labeled by σ. If σ^* appears twice in R, let $\sigma = \sigma_i = \sigma_j$ with $i < j$. We construct a weighted graph G'_σ by adding to G_σ the edge (v_i, v_{j-1}) with a large cost. Now we need to find a least-cost path in G'_σ from v_{i-1} to v_j in which every edge appears at least once, and the new edge (v_i, v_{j-1}) appears exactly once. Since the edge (v_i, v_{j-1}) has a large cost, it can be done simply by finding an optimal postman path from v_{i-1} to v_j in G'_σ. Finally, as in Theorem 9, we construct a path P by concatenating the corresponding paths for every block b_i in R. In Appendix A.8 we describe how to handle the case where R contains single-letter blocks. ◀

The case of 1-diverse chain REs follows similar considerations and applies the ideas used in the proof of Theorem 7 in the case of the CEP problem. The proof can be found in Appendix A.9.

▶ **Theorem 13.** *The CCP problem can be solved in polynomial time for specifications given by a 1-diverse chain RE with a fixed number of blocks.*

8 Eulerian Languages

The *Eulerian language* of a Σ-labeled graph G, denoted $EL(G)$, is the set of words read along Eulerian paths in G. Formally, $EL(G) = \{l(P) \in \Sigma^* : P \text{ is an EP in } G\}$. Clearly, the *nonemptiness problem*, namely deciding whether $EL(G) \neq \emptyset$, coincides with the EP problem and can thus be solved in polynomial time. Given a regular language $L \subseteq \Sigma^*$, the *satisfaction problem* for G and L is to decide whether $EL(G) \cap L \neq \emptyset$. It is easy to see that the satisfaction problem coincides with the CEP problem, and is thus NP-complete (Theorem 3). Given a word $w \in \Sigma^*$, the *membership problem* for G and w is to decide whether $w \in EL(G)$. By Theorem 6, the CEP problem is NP-complete also for singleton specifications, implying that so is the membership problem.

In this section we study additional problems about the Eulerian language of G. Problems that compare it with other languages, given by an NFA, DFA, or RE, or given as the Eulerian language of another graph. Not all our complexities are tight, but we are able to place all problems in different levels of the polynomial hierarchy.

▶ **Theorem 14.** *Consider a labeled graph G and a specification L given by an NFA, DFA, or RE. Deciding whether $EL(G) \subseteq L$ is co-NP-complete. Furthermore, it is co-NP-hard already for a fixed-size specification.*

Proof. For the upper bound, note that a witness for $EL(G) \not\subseteq L$, namely an EP in G that does not satisfy L, can be verified in polynomial time. For the lower bound, recall that the CEP problem is NP-hard already for fixed-size specifications (Theorem 5). Observe that there is an EP that satisfies L iff there is an EP that does not satisfy $\Sigma^* \setminus L$. Since L is given by a fixed-size NFA, DFA, or RE, the size of an NFA, DFA, or RE for its complement $\Sigma^* \setminus L$ is also fixed, and we are done. ◀

▶ **Theorem 15.** *Consider a labeled graph G and a specification L given by an NFA, DFA, or RE. Deciding whether $L \subseteq EL(G)$ is in Π_2^p and is NP-hard.*

Proof. The lower bound follows from the NP-hardness of the membership problem. The upper bound follows from the fact that deciding whether $L \not\subseteq EL(G)$ can be done with a nondeterministic polynomial-time Turing machine that uses an oracle for the membership problem. ◀

▶ **Theorem 16.** *Consider two labeled graphs G and G'. Deciding whether $EL(G') \subseteq EL(G)$ and deciding whether $EL(G') \cap EL(G) \neq \emptyset$ is in Π_2^p and Σ_2^p respectively, and is NP-hard.*

Proof. For the lower bound, we show a reduction from the membership problem. Given a word w and a graph G, we construct a graph G' such that $EL(G') = \{w\}$. Now, $w \in EL(G)$ iff $EL(G') \subseteq EL(G)$ iff $EL(G') \cap EL(G) \neq \emptyset$. For the upper bound, observe that deciding whether $EL(G') \not\subseteq EL(G)$ and whether $EL(G') \cap EL(G) \neq \emptyset$ can be done with a nondeterministic polynomial-time Turing machine that uses an oracle for the membership problem. ◀

Since $EL(G)$ contains only words of a fixed length, we know that $EL(G)$ is finite and hence regular. On the other hand, given a regular language $L \subseteq \Sigma^*$, even one all whose words are of the same length, it is not clear whether there is a graph G such that $EL(G) = L$. For example, it is possible to find a two-state labeled graph G such that $EL(G) = \{abcd, adbc, cbad, cdba\}$ (the reader is encouraged to search for it. See a hint in Appendix A.10), but it is impossible to add $abdc$ to the Eulerian language. An upper bound for the problem follows from our ability to bound the number of edges in the candidate graph G. The tight complexity, however, is still open.

▶ **Theorem 17.** *For a language L given by an NFA, DFA, or RE, deciding whether there is a labeled graph G such that $EL(G) = L$ is in Σ_3^p.*

Proof. Follows from the fact that it can be done by a nondeterministic polynomial-time Turing machine with oracles for the problems described in Theorems 14 and 15. ◀

References

1 S. Abiteboul and V. Vianu. Regular path queries with constraints. *J. Comput. Syst. Sci.*, 58(3):428–452, 1999.

2 E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM J. Comput.*, 38(4):1602–1623, 2008.

3 G. Avni, O. Kupferman, and T. Tamir. Network-formation games with regular objectives. In *Proc. 17th Int. Conf. on Foundations of Software Science and Computation Structures*, volume 8412 of *Lecture Notes in Computer Science*, pages 119–133. Springer, 2014.

4 C. Barrett, R. Jacob, and M. Marathe. Formal-language-constrained path problems. *SIAM Journal on Computing*, 30(3):809–837, 2000.

5 V. Blue, J. Adler, and G. List. Real-time multiple-objective path search for in-vehicle route guidance systems. *Journal of the Transportation Research Board*, 1588:10–17, 1997.

6 P.G. Bradford and D.A. Thomas. Labeled shortest paths in digraphs with negative and positive edge weights. *RAIRO-Theoretical Informatics and Applications*, 43(03):567–583, 2009.

7 A.L. Buchsbaum, P.C. Kanellakis, and J.S. Vitter. A data structure for arc insertion and regular path finding. *Annals of Mathematics and Artificial Intelligence*, 3(2-4):187–210, 1991.

8 D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Reasoning on regular path queries. *ACM SIGMOD Record*, 32(4):83–92, 2003.

9 W. Cheng and M. Pedram. Power-optimal encoding for a DRAM address bus. *IEEE Trans. VLSI Syst.*, 10(2):109–118, 2002.

10 T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

11 M. Dror, H. Stern, and P. Trudeau. Postman tour on a graph with precedence relation on arcs. *Networks*, 17(3):283–294, 1987.

12 H.A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part i: The chinese postman problem. *Operations Research*, 43(2):231–242, 1995.

13 L.R. Ford and D.R. Fulkerson. *Flows in networks*. Princeton Univ. Press, Princeton, 1962.

14 S. J. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:11–121, 1980.

15 M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. Freeman and Co., 1979.

16 G. Ghiani and G. Improta. An algorithm for the hierarchical chinese postman problem. *Operations Research Letters*, 26(1):27–32, 2000.

17 S. Hannenhalli, W. Feldman, H.F. Lewis, S.S. Skiena, and P.A. Pevzner. Positional sequencing by hybridization. *Computer applications in the biosciences: CABIOS*, 12(1):19–24, 1996.

18 T. Ibaraki and S. Poljak. Weak three-linking in eulerian digraphs. *SIAM journal on Discrete Mathematics*, 4(1):84–98, 1991.

19 J. Kari. Synchronizing finite automata on Eulerian digraphs. *Theoretical Computer Science*, 295:223–232, 2003.

20 K.I. Kawarabayashi, Y. Kobayashi, and B. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012.

21 H.L.M. Kerivin, M. Lacroix, and A.R. Mahjoub. Models for the single-vehicle preemptive pickup and delivery problem. *Journal of Combinatorial Optimization*, 23(2):196–223, 2012.

22 H.L.M. Kerivin, M. Lacroix, and A.R. Mahjoub. On the complexity of the Eulerian closed walk with precedence path constraints problem. *Theoretical Computer Science*, 439:16–29, 2012.

23 P. Korteweg and T. Volgenant. On the hierarchical chinese postman problem with linear ordered classes. *European Journal of Operational Research*, 169(1):41–52, 2006.

24 O. Kupferman and T. Tamir. Properties and utilization of capacitated automata. In *Proc. 34th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 29 of *LIPIcs*, pages 33–44. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2014.

25 A.O. Mendelzon and P.T. Wood. Finding regular simple paths in graph databases. *SIAM Journal on Computing*, 24(6):1235–1258, 1995.

26 E. Moreno and M. Matamala. Minimal Eulerian circuit in a labeled digraph. In *LATIN 2006: Theoretical Informatics*, pages 737–744. Springer, 2006.

27 G. Naves and A. Sebő. Multiflow feasibility: an annotated tableau. In *Research Trends in Combinatorial Optimization*, pages 261–283. Springer, 2009.

28 P.A. Pevzner, H. Tang, and M.S. Waterman. An Eulerian path approach to dna fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001.

29 N. Robertson and P.D. Seymour. Graph minors. xiii. the disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995.

30 J. Vygen. Disjoint paths. report no. 94816. *Research Institute for Discrete Mathematics, University of Bonn*, 1994.

A Proofs

A.1 Proof of Theorem 4

Given a graph $G = \langle V, E\rangle$, we construct a simple labeled graph G'' and a DFA $\mathcal{A}$ such that there is a Hamiltonian path in G iff there is an EP in G'' that satisfies $L(\mathcal{A})$. The idea is similar to the construction of G' in the proof of Theorem 3, except that we add a new letter $\#$ to the alphabet V and replace every (parallel) self-loop labeled v in G' by a (disjoint) cycle labeled $v\#$ in G''. Formally, $G'' = \langle V \cup \{u\}, (\{u\} \times V) \cup (V \times \{u\}), l\rangle$ is over the alphabet $\Sigma = V \cup \{\#\}$ and is such that for every $v_i \in V$, we have $l((u, v_i)) = v_i$ and $l((v_i, u)) = \#$. See Figure 2 for an example. That is, G''

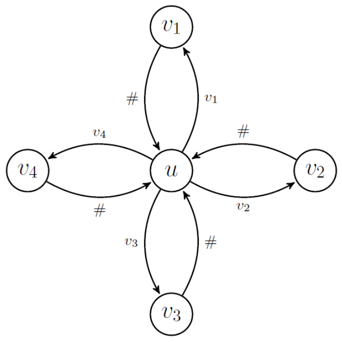

Figure 2 A graph G'' for G with $V = \{v_1, v_2, v_3, v_4\}$.

is “star shaped" and as has been the case with G', its EPs correspond to permutations of V.

Now, we define the specifications DFA $\mathcal{A}$ so that $L(\mathcal{A})$ includes exactly all words $w \in \Sigma^*$ such that the projection of w on V induces a path in G. In other words, $\mathcal{A}$ accepts w iff w is of the form $\#^* v_1 \#^* v_2 \ldots v_n \#^*$, for a path $v_1, v_2, \ldots, v_n$ in G. It is easy to define $\mathcal{A}$ as above by adding to G self-loops labeled $\#$ and an initial state that has a transition to all states. Formally, $\mathcal{A} = \langle \Sigma, V \cup \{s\}, \{s\}, \delta, V\rangle$, where $\delta(v_i, v_j) = \{v_j\}$ for every $(v_i, v_j) \in E$, $\delta(s, v_i) = \{v_i\}$ for every $v_i \in V$, and $\delta(v, \#) = \{v\}$ for every $v \in V \cup \{s\}$. Now, there is a Hamiltonian path in G iff there is a permutation of V that forms a path in G iff there is an EP in G'' that satisfies $L(\mathcal{A})$.

A.2 Correctness of the reduction in the proof of Theorem 5

Assume first that G has a Hamiltonian path P from a vertex s to a vertex t, we show that G' has an Eulerian cycle that satisfies $L(R)$. The Eulerian cycle starts with the path Q in G' that corresponds to P (starting from the vertex s_{in} and ending in t_{out}). The word $l(Q)$ is of the form $a(ba)^*$. Now, after we remove the edges of Q from G' the in degree of every vertex except for s_{in} and t_{out} is still equal to its out degree. Also, the graph G' is still strongly connected because of the edges in E_5 and in E_6. Therefore, after removing the edges of Q from G', the graph has an EP Q' from t_{out} to s_{in}. Since Q already used all the edges labeled by a, then $l(Q')$ is of the form $(b+c)^*$. Therefore, the path obtained by concatenating Q and Q' is an Eulerian cycle in G' of the form $a(ba)^*(b+c)^*$ and therefore satisfies $L(R)$.

Assume now that G' has an EP P that satisfies $L(R)$, we show that G has a Hamiltonian path. Since P is an EP and $l(P)$ is of the form $(a+b)^*(b+c)^*$, then it starts by visiting every edge labeled by a exactly once and using only edges labeled by b in between. Thus, it starts with a path in G' that corresponds to a Hamiltonian path in G.

A.3 Correctness of the reduction in the proof of Theorem 6

We detail the second direction of the proof. Assume that there is a Hamiltonian path P in G from v^1 to v^n. We construct an Eulerian cycle in G' as follows: The Eulerian cycle starts with the path Q in G' corresponding to P. This path starts with the edge (v^1_{in}, v^1_{out}), ends with the edge (v^n_{in}, v^n_{out}) and visits the edge (v^i_{in}, v^i_{out}) for every i exactly once. Note that $l(Q) = a(ba)^{n-1}$. Let $Q = e_1, \ldots, e_{2n-1}$. For an edge $e = (u, v)$, we denote $e' = (v, u)$, that is, e' is the edge in the opposite direction to e. The Eulerian cycle continues with the path $Q' = e'_{2n-1}, \ldots, e'_1$, that is, it returns from the vertex v^n_{out} to the vertex v^1_{in}. Note that $l(Q') = c^{2n-1}$. Let $E' = \mathcal{E} \setminus (Q \cup Q')$ be the set of edges that are left after removing the edges of Q and Q' from G'. Note that for every $e \in E'$ we have $l(e) \in \{b, c\}$, and that $e \in E'$ with $l(e) = b$ iff $e' \in E'$ with $l(e') = c$. We denote by d_i the in degree in G of the vertex v^i, for every $1 \le i \le n$. The Eulerian cycle now visits all the edges incident to v^1_{in} except for (v^1_{in}, w) and (w, v^1_{in}), by visiting an outgoing edge e and then visiting the edge e'. We denote this path by Q_1 and note that $l(Q_1) = (cb)^{d_1}$. Then, the Eulerian cycle visits the edges $(v^1_{in}, w), (w, v^2_{in})$ (labeled by cb) in order to move to the vertex v^2_{in}. The Eulerian cycle now visits all the edges incident to v^2_{in}, except for (v^2_{in}, w) and (w, v^2_{in})), with a path Q_2 such that $l(Q_2) = (cb)^{d_2-1}$. Note that v^2_{in} had only $d_2 - 1$ such pairs of edges since the paths Q and Q' have already used one such pair. The Eulerian cycle now moves to v^3_{in} through w and it continues similarly until it reaches the vertex v^n_{in} and its incident edges. The Eulerian cycle continues with the path $Q'' = (v^n_{in}, w), (w, v^1_{out}), (v^1_{out}, w), (w, v^2_{out}), (v^2_{out}, w), (w, v^3_{out}), \ldots, (v^n_{out}, w), (w, v^1_{in})$. Note that $l(Q'') = (cb)^{n+1}$ and that it concludes the construction of the required Eulerian cycle.

A.4 Proof of Lemma 8

Assume that G has an EP from s to t (otherwise, which can be detected in polynomial time, the algorithm returns a negative answer). Let $e_i = (u_i, v_i)$ for all $1 \le i \le k$, and let $G' = \langle V, E \setminus \{e_1, \ldots, e_k\}\rangle$ be the graph obtained from G by removing the edges $e_1, \ldots, e_k$. Note that the problem of finding an EP in G from s to t that respects τ is equivalent to the problem of finding the following edge-disjoint paths in G': from s to u_1, from v_1 to u_2, …, from v_{k-1} to u_k, and from v_k to t. Indeed, an EP from s to t in G such that $e_1 \prec \ldots \prec e_k$ induces the above edge-disjoint paths. Conversely, the above edge-disjoint paths in G' can be extended to an EP in G: First, we connect these paths by adding the edges $e_1, \ldots, e_k$. This results in a path P from s to t in which no edge appears more than once. Now, since we assume that there is an EP in G from s to t, we can proceed as in the standard algorithm for finding an EP, and complete P to an EP from s to t. Thus, the problem is reduced to an EDP problem in G'.

Let G'_D be the demand graph for the EDP problem in G' described above. Note that for every vertex in the graph $G' + G'_D$ the in degree is equal to the out degree. Thus, the graph $G' + G'_D$

is either Eulerian (if it is connected) or a union of disjoint Eulerian subgraphs. The case where G' is not connected is easier, since in this case the edge-disjoint paths problem in G' can be solved independently for each connected component. Therefore, we assume that G' is connected and thus that $G' + G'_D$ is Eulerian. Since $G' + G'_D$ is Eulerian, the edge-disjoint paths problem for three paths in G' has a polynomial-time algorithm, and therefore the problem of finding an EP from s to t with $e_1 \prec e_2$ in G has a polynomial-time algorithm.

Now, assume that there is a polynomial-time algorithm for finding an EP with $e_1 \prec \ldots \prec e_k$ for $k > 2$ in a directed graph, we show an algorithm for the EDP problem for $k+1$ paths with an Eulerian promise on the demand. Let $H = \langle V_H, E_H \rangle$ be a directed graph, let $\{(s_i, t_i) : 1 \leq i \leq k+1\}$ be the edge-disjoint paths demand and let H_D be the corresponding demand graph. Assume that $H + H_D$ is Eulerian. Also, we can assume that H is connected, as if H is not connected but $H + H_D$ is connected, then there is a demand from some vertex to a vertex in another connected component in H. Let $H' = \langle V_H, E_H \cup \{(t_i, s_{i+1}) : i = 1 \ldots k\} \rangle$. Note that H' has an EP from s_1 to t_{k+1}. Now, we use the polynomial-time algorithm for finding an EP in H' from s_1 to t_{k+1} such that $(t_1, s_2) \prec \ldots \prec (t_k, s_{k+1})$. If such an EP exists, then it induces $k+1$ edge-disjoint paths in H as required. Conversely, if there are $k+1$ edge-disjoint paths in H as required, then the required EP in H' exists, and can be obtained by connecting the edge-disjoint paths with the edges (t_i, s_{i+1}) for $i = 1, \ldots, k$ and extending the resulting path in the standard way.

A.5 Proof of claim in the proof of Theorem 9

We claim that there is an EP in G'_σ from v_{i_1-1} to v_{i_m} such that $e_{i_1} \prec e_{i_2} \prec \ldots \prec e_{i_{m-1}}$ iff the required EP in G exists. If the required EP in G exists then by taking the subpaths corresponding to $b_{i_1}, \ldots, b_{i_m}$ and connecting them with the edges $e_{i_1}, \ldots, e_{i_{m-1}}$ we obtain the required EP in G'_σ. Conversely, if for every $\sigma \in \Sigma$ the above EP in G'_σ exists, then the required EP P in G can be obtained as follows: For every block b_i we append to P a subpath. If b_i is a single-letter block then we add the single-edge subpath (v_{i-1}, v_i). If $b_i = \sigma^*$ for some $\sigma \in \Sigma$, we append to P the corresponding subpath of the EP in G'_σ. This subpath starts in v_{i-1} and ends in v_i. Note that P is an EP in G that satisfies $L(R)$.

A.6 Proof of Theorem 10

Assume first that G has a Hamiltonian path P from a vertex s to a vertex t, we show that G' has an Eulerian cycle that satisfies $L(R)$. The Eulerian cycle starts with the path Q in G' that corresponds to P (starting from the vertex s_{in} and ending in t_{out}). Since Q uses only edges from $E_1 \cup E_2$, then $l(Q)$ contains a word of the form a^*. Now, after we remove the edges of Q from G' the in degree of every vertex except for s_{in} and t_{out} is still equal to its out degree. Also, the graph G' is still strongly connected because of the edges in E_5 and in E_6. Therefore, after removing the edges of Q from G', the graph has an EP Q' from t_{out} to s_{in}. Since Q already used all the edges in E_1, then $l(Q')$ contains a word of the form b^*. Therefore, the path obtained by concatenating Q and Q' is an Eulerian cycle in G' that satisfies $L(R)$.

Assume now that G' has an EP P that satisfies $L(R)$, we show that G has a Hamiltonian path. Since P is an EP and $l(P)$ contains a word of the form a^*b^*, then it starts by visiting every edge in E_1 exactly once and using only edges from E_2 in between. Thus, it starts with a path in G' that corresponds to a Hamiltonian path in G.

A.7 A polynomial-time algorithm for the Chinese postman problem

Note that a graph has a postman cycle iff it is strongly connected. Let G be a strongly connected weighted graph. If G has an Eulerian cycle then it is also an optimal postman cycle. Otherwise, G

has vertices with unbalanced degrees, that is, their in degrees and out degrees are not equal. In this case, we construct a graph G' by adding to G paths from vertices with an in degree greater than their out degree to those with an out degree greater than their in degree, such that they would make the in degree of every vertex equal to its out degree. We choose the paths such that the set of edges that are added to G is of minimal cost. This can be done by solving a network-flow problem. Then, the optimal postman cycle is obtained by finding an Eulerian cycle in G'. In order to find an optimal postman path from a vertex s to a vertex t in a weighted graph H, we first add to H a new edge (t, s) with a large cost and then find an optimal postman cycle in the resulting graph. If there is a postman path from s to t then the edge (t, s) appears only once in the optimal postman cycle since it has a large cost.

A.8 Handling the case of single-letter blocks in Theorem 12

We handle the case where R contains single-letter blocks. For every i such that $b_i = \sigma$ for some $\sigma \in \Sigma$, the graph G must have an edge $e_i = (v_{i-1}, v_i)$ such that $l(e_i) = \sigma$. If there are parallel edges (v_{i-1}, v_i) that are labeled by σ then we choose e_i to be an edge that has not been chosen yet, and if all of these parallel edges have already been chosen, we choose e_i to be the least-cost edge. Now, assume that the expression σ^* also appears in R in two blocks (the case where it appears only in one block is similar). We find a path in G'_σ as in the standard algorithm, but now this path does not have to include edges that have been chosen for the single-letter blocks. Since there is a fixed number of single-letter blocks, we can run over all the subsets of these edges, and for each subset remove the edges from G'_σ and find the optimal postman path as in the standard algorithm.

A.9 Proof of Theorem 13

Let $G = \langle V, E, l, c\rangle$ be a labeled weighted graph and let k be a fixed number. Let $R = b_1 \ldots b_k$ be a RE, where for every i we have $b_i = w_i^*$ or $b_i = w_i$ for some $w_i \in \Sigma^*$, and every $\sigma \in \Sigma$ appears at most once in R. We assume that every letter that appears in G, appears also in R, because otherwise the CCP problem is trivial. As in the proof of Theorem 12, we run over all the options for choosing $k + 1$ vertices $v_0, \ldots, v_k \in V$, and find a least-cost path in G from v_0 to v_k that satisfies $L(R)$, contains all the edges and can be partitioned into k subpaths, such that the i-th subpath starts in v_{i-1}, ends in v_i and corresponds to b_i.

Let $G_i = \langle V_i, E_i, l, c\rangle$ be the subgraph of G induced by the edges labeled by letters in w_i. We show how to find a least-cost path in G_i from v_{i-1} to v_i that contains all the edges of G_i and corresponds to b_i. If $b_i = w_i$, then we check whether G_i contains every letter in w_i exactly once, and whether w_i induces a path in G_i from the vertex v_{i-1} to the vertex v_i. Assume now that $b_i = w_i^*$, and $w_i = \sigma_0 \ldots \sigma_{n-1}$. As in the proof of Theorem 7, we construct a weighted graph $G'_i = \langle V'_i, E'_i, c'\rangle$, where $V'_i = \{\langle v, j\rangle : v \in V_i \text{ and } 0 \leq j \leq n-1\}$ and $E'_i = \{(\langle u, j\rangle, \langle v, j+1\ (mod\ n)\rangle) : (u, v) \in E_i \text{ and } l((u, v)) = \sigma_j\}$ and $c'((\langle u, j\rangle, \langle v, j+1\ (mod\ n)\rangle)) = c((u, v))$. That is, G'_i includes n copies of G_i such that every edge (u, v) with $l((u, v)) = \sigma_j$ in G_i induces an edge with the same weight from u in the j-th copy to v in the $(j + 1)$-th copy in G'_i. Note that finding a least-cost path in G_i from v_{i-1} to v_i that contains all the edges of G_i and corresponds to b_i can be done by finding an optimal postman path in G'_i from $\langle v_{i-1}, 0\rangle$ to $\langle v_i, 0\rangle$.

A.10 An example of an Eulerian language of a labeled graph

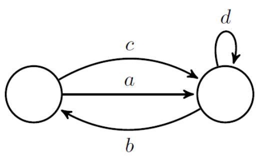

Figure 3 A labeled graph G such that $EL(G) = \{abcd, adbc, cbad, cdba\}$.