On the Parameterized Complexity of Edge-Linked Paths

Neeldhara Misra[1], Fahad Panolan[2(✉)], and Saket Saurabh[3]

[1] Indian Institute of Technology, Gandhinagar, Gandhinagar, India
neeldhara.m@iitgn.ac.in
[2] Department of Informatics, University of Bergen, Bergen, Norway
fahad.panolan@ii.uib.no
[3] The Institute of Mathematical Sciences, HBNI, Chennai, India
saket@imsc.res.in

Abstract. An edge Hamiltonian path of a graph is a permutation of its edge set where every pair of consecutive edges have a vertex in common. Unlike the seemingly related problem of finding an Eulerian walk, the edge Hamiltonian path is known to be a NP-hard problem, even on fairly restricted classes of graphs. We introduce a natural optimization variant of the notion of an edge Hamiltonian path, which seeks the longest sequence of distinct edges with the property that every consecutive pair of them has a vertex in common. We call such a sequence of edges an edge-linked path, and study the parameterized complexity of the problem of finding edge-linked paths with at least k edges. We show that the problem is FPT when parameterized by k, and unlikely to admit a polynomial kernel even on connected graphs.

On the other hand, we show that the problem admits a Turing kernel of polynomial size. To the best of our knowledge, this is the first problem on general graphs to admit Turing kernels with adaptive oracles (for which a non-adaptive kernel is not known). We also design a single-exponential parameterized algorithm for the problem when parameterized by the treewidth of the input graph.

Keywords: FPT · Turing kernelization · Edge Hamiltonian cycle

1 Introduction

An edge Hamiltonian path of a graph is a permutation of its edge set where every pair of consecutive edges have a vertex in common. The notion is a classical one, well-studied in the context of structural graph theory, and more recently, has received attention from the computational perspective as well. Unlike the seemingly related problem of finding an Eulerian walk, the edge Hamiltonian

This work is supported by the European Research Council (ERC) via grant LOPPRE, reference 819416 and Norwegian Research Council via project MULTIVAL.

R. van Bevern and G. Kucherov (Eds.): CSR 2019, LNCS 11532, pp. 286–298, 2019.
https://doi.org/10.1007/978-3-030-19955-5_25

path is known to be a NP-hard problem, even on fairly restricted classes of graphs such as bipartite graphs and graphs of maximum degree three [1,11,13].

We introduce a natural optimization variant of the notion of an edge Hamiltonian path, which is the following: what is the longest sequence of distinct edges with the property that every consecutive pair of them has a vertex in common? Note that this subsumes the question of finding an edge Hamiltonian path as a special case, and therefore the classical hardness of the problem follows immediately. For ease of discussion, we use the phrase "edge-linked paths" to refer to sequences of distinct edges that have the property of every consecutive pair of edges having a common vertex between them. We use the abbreviation LELP to refer to the problem of finding the longest edge-linked path.

It turns out that studying LELP from a parameterized perspective leads to interesting new algorithms for the problem. There are two natural parameters that emerge for LELP: the first is the standard parameter, which is the length of the edge sequence, and the second is the treewidth of the input graph. The treewidth parameter turns out to be useful also in the context of the original edge Hamiltonian path problem, see, for instance [12]. In this contribution, we establish the following results:

- LELP is FPT when parameterized by either the standard parameter or the treewidth. In particular, for the treewidth parameter, we demonstrate a single exponential algorithm for this problem.
- We argue that the problem is unlikely to admit a polynomial kernel with the standard parameter, even when the graph is connected. On the other hand, we show that the problem does admit a Turing kernel of polynomial size.

We remark here that our demonstration of a Turing kernel is particularly interesting, since our algorithm is based on oracle queries made in an adaptive fashion. Recall that a *Turing kernel* is a polynomial time algorithm that can solve the problem with access to an oracle for the problem, operating under the constraint that the oracle can only answer queries for small instances. The size of the largest instance on which we invoke the oracle is the "size" of the Turing kernel. Turing kernels are central to the study of problems that are unlikely to admit polynomial kernels, and yet, they have been demonstrated for only a small number of problems. Further, most Turing kernels work by producing polynomially many instances of bounded size without using the oracle at all[1]. Indeed, only a few Turing kernels are known that take full advantage of the oracle framework. These include the problems of finding long paths and cycles on restricted classes of graphs [9,10] and the weighted independent set problem on bull-free graphs [14]. Our contribution adds a natural problem to this limited list of problems known to admit Turing kernels using oracles in an adaptive fashion. To the best of our knowledge LELP is the first problem on *general*

[1] These instances typically have the property that the input instance is a Yes-instance if and only if one of these instances is a Yes-instance: therefore, the oracle can be applied to each instance in turn to solve the problem.

graphs to admit Turing kernels using adaptive oracles (for which a non-adaptive kernel is not known).

Apart from the standard parameter and treewidth, we also consider a natural "above-guarantee" parameter. Note that the maximum degree of a graph provides an easy lower bound on the length of a longest edge-linked path, since one has the sequence of edges incident on the vertex with the largest degree. Therefore, an interesting question to ask is if there is an edge-linked path of length at least $\Delta(G) + k'$, where $\Delta(G)$ denotes the maximum degree of G. This is an example of an *above-guarantee parameter*, where the parameter signifies the length of the path that is possible beyond what is structurally guaranteed to exist. We show that LELP is also FPT by this parameterization.

Methodology. It is quite straightforward to observe that LELP is FPT when parameterized by the standard parameter k. One method is to find a path of length k in the line graph of the input graph. Another approach, for instance, would be to observe that we can say YES if either the depth of a DFS traversal exceeds k, or if there is a vertex of degree at least k, since any usual path is also an edge-linked path, and a vertex of degree k naturally corresponds to an edge-linked path of length k. If both of these cases do not arise, then the size of the graph is easily seen to be bounded as a function of k. In fact it is known that if the depth of the DFS tree is at most k, then the treewidth of the graph is at most $k-1$. We propose an efficient algorithm parameterized by treewidth, using techniques based on representative families. In particular, our main result in this context is the following.

Theorem 1. *Let G be an n-vertex graph given together with its tree decomposition of width* $\mathbf{tw}$*. Then* LELP *can be solved in time* $\mathcal{O}\Big(\left(1+2^{(\omega+3)}\right)^{\mathbf{tw}} \mathbf{tw}^{\mathcal{O}(1)} \cdot nm\Big)$ *where $m = |E(G)|$, and ω is the matrix-multiplication exponent.*

In the context of kernelization, we begin by observing that the problem is unlikely to admit a polynomial kernel with the standard parameter, by a standard application of the disjoint union construction. However, the graphs obtained by this construction are not connected, so we establish, using an explicit cross-composition, the hardness of kernelization for connected graphs. On the other hand, we also establish a polynomial-size Turing kernel for the problem. The algorithm we propose makes use of Tutte decompositions, which are tree decompositions that have additional properties—most notably that the torsos of the bags are 3-edge-connected. In this context, we are able to exploit the fact that such graphs are known to admit large Eulerian subgraphs [2], which imply the existence of long edge-linked paths. Therefore, if the Tutte decomposition of the given graph has a large bag, then we already have a Yes-instance on our hands. Otherwise, it turns out that we can use the decomposition to find a separation (A, B) of order at most two where one of A or B has bounded size.

From this point, the algorithm is based on careful invocations of an oracle that can find long edge-linked paths on the smaller side of the separation, either

to determine that we have a Yes-instance or to discover a vertex that can be safely removed, thus making progress at every step. This involves a careful analysis of all possible ways in which a solution can split across the separation. The overall approach that we employ is inspired by the techniques used for obtaining Turing kernels for the problem of finding long paths and cycles on special classes of graphs [10].

Theorem 2. LELP*, when parameterized by* k*, does not admit a polynomial kernel on general graphs and connected graphs, unless* $\mathsf{CoNP} \subseteq \mathsf{NP/poly}$*. On the other hand, the problem admits a Turing kernel with* $O(k^{3.656})$ *vertices and* $O(k^{4.656})$ *edges.*

Finally, for parameterized "above maximum degree", we show an FPT algorithm obtained by relying on bounding the treewidth of certain connected components in the graph.[2]

Related Work. The Edge Hamiltonian Path is known to be NP-complete even on bipartite graphs or graphs with maximum degree 3 [1,11,13]. Demaine et al. [6] presented an XP (i.e. running in time $n^{f(k)}$) algorithm for Edge Hamiltonian Path on bipartite graphs, where k is the size of the smaller part and asked whether it can be improved to an FPT algorithm. Lampis et al. [12] answered this question affirmatively by giving a cubic edge kernel for Edge Hamiltonian Path when parameterized by vertex cover. They also show that it is FPT on hypergraphs when parameterized by the size of a hitting set. They also studied the problem with parameters treewidth and clique-width of the input graph. The running times of their algorithms when parameterized by treewidth and clique width are $\mathbf{tw}^{\mathcal{O}(\mathbf{tw})}n^{\mathcal{O}(1)}$ and $\mathbf{cw}^{\mathcal{O}(\mathbf{cw}^2)}n^{\mathcal{O}(1)}$, respectively, where $\mathbf{tw}, \mathbf{cw}$ and n are the treewidth, clique-width and number of vertices of the input graph. To the best of our knowledge, this is the first study that uses the length of an edge-linked path as a parameter. We also note LELP can be solved also by using known algorithms for Long Path on the line graph of the input graph. However, in the context of the standard parameter, this yields no insight into the kernelization complexity. Also, when parameterized by treewidth, this approach does not give us FPT algorithms: note that the treewidth of the line graph can be arbitrarily larger than the treewidth of the input graph: for instance, note that the line graph of a star (treewidth one) is a clique (treewidth n).

2 Preliminaries

We refer the reader to [4,7] for standard terminology and notions in graph theory and parameterized algorithms. Unless made explicit, we use standard notation throughout. We introduce and recall some important definitions below.

[2] Due to lack of space, the algorithm parameterized by treewidth and the arguments for the above-guarantee parameter are deferred to the full version of the paper.

For a graph $G, S \subseteq V(G)$, and $F \subseteq E(G)$, we use $E^G_{in}(S)$ to denote the set of edges incident to vertices in S, and $V(F)$ to denote the set of end-vertices of F. When the graph is clear from the context we use $E_{in}(S)$ instead of $E^G_{in}(S)$. For a graph G, a pair (A, B), where $A, B \subseteq V(G)$ and $A \cup B = V(G)$ is called a *separation* of G if there is no edge in G with one endpoint in $A \setminus B$ and the other in $B \setminus A$. The *order* of the separation (A, B) is $|A \cap B|$. A minimal separator in a connected graph G is an inclusion minimal vertex set $S \subseteq V(G)$ such that $G - S$ is disconnected. A vertex set of a disconnected graph is a minimal separator if it is a minimal separator for one of its connected components. For a vertex set U, define $\mathsf{torso}(G, U)$ as the graph obtained from $G[U]$ by adding an edge between each pair of vertices in U that are connected by a path in G whose internal vertices are not from U. Following simple observation is used many times in the paper.

Observation 1. *The number of odd degree vertices in a graph is even.*

Definition 1. *A* tree-decomposition *of a graph G is a pair $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$, where $\mathbb{T}$ is a rooted tree, such that (i) $\bigcup_{t \in V(\mathbb{T})} X_t = V(G)$, (ii) for every edge $xy \in E(G)$ there is a $t \in V(\mathbb{T})$ with $\{x, y\} \subseteq X_t$, and (iii) for every vertex $v \in V(G)$ the subgraph of $\mathbb{T}$ induced by the set $\{t \mid v \in X_t\}$ is connected.*

The width *of a tree decomposition is $\max_{t \in V(\mathbb{T})} |X_t| - 1$ and the* treewidth *of G is the minimum width over all tree decompositions of G and is denoted by $\mathbf{tw}(G)$. The* adhesion *of an edge $\{t, t'\} \in E(\mathbb{T})$ is $|X_t \cap X_{t'}|$. The* adhesion *of a tree decomposition is the maximum adhesion of an edge in $\mathbb{T}$.*

Proposition 1. *For every graph G, there is a tree decomposition $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ of adhesion at most two, called a* Tutte decomposition*, such that the following conditions hold.*

- *For each node $t \in V(\mathbb{T})$, the graph $\mathsf{torso}(G, X_t)$ is a 3-vertex-connected topological minor of G.*
- *For each edge $\{t, t'\} \in E(\mathbb{T})$, either $X_t \cap X_{t'} = \emptyset$ or $X_t \cap X_{t'}$ is a minimal separator in G.*

Proposition 2 (Hopcroft and Tarjan [8]). *There is a linear time algorithm to compute a Tutte decomposition of a given graph.*

Recall that an edge-linked path is a sequence of distinct edges $e_1, e_2, \ldots, e_k$ such that every consecutive pair of edges in the sequence have a vertex in common. The length of an edge-linked path is the number of edges that belong to the path. We introduce the following problem:

Long Edge-Linked Paths	Parameter: k/tw
Input: A graph G and an integer k	
Question: Does G have an edge-linked path of length at least k?	

We denote by LELP(k) the problem of finding an edge-linked path of length at least k, parameterized by k. We will need the following result from [2]. In the following statement, following the notation of [2], we use K_2^3 to denote the graph with two vertices joined by three parallel edges. Further, a graph is said to be *Eulerian* if it is connected and all its vertices have even degree.

Theorem 3 ([2]). *Let G be a 3-edge-connected graph, $e, f \in E(G)$, and assume $G \neq K_2^3$. Then G contains an Eulerian subgraph H such that $e, f \in E(H)$ and $|E(H)| \geq (|E(G)|/6)^\alpha + 2$, where $\alpha \approx 0.753$ is the real root of $4^{1/x} - 3^{1/x} = 2$.*

Parameterized Complexity. We refer to [4] for a detailed introduction to parameterized complexity. We provide some key definitions pertinent to our arguments below.

Definition 2 (polynomial equivalence relation [3]). *An equivalence relation $\mathcal{R}$ on Σ^*, where Σ is a finite alphabet, is called a* polynomial equivalence relation *if the following holds:* (1) *equivalence of any $x, y \in \Sigma^*$ can be checked in time polynomial in $|x| + |y|$, and* (2) *any finite set $S \subseteq \Sigma^*$ has at most $(\max_{x \in S} |x|)^{\mathcal{O}(1)}$ equivalence classes.*

Definition 3 (cross-composition [3]). *Let $L \subseteq \Sigma^*$ and $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that L* cross-composes *into Q if there is a polynomial equivalence relation $\mathcal{R}$ on Σ^* and an algorithm which given t strings $x_1, \ldots, x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that: (i) $(x^*, k^*) \in Q \Leftrightarrow x_i \in L$ for some $1 \leq i \leq t$ and (ii) k^* is bounded by a polynomial in $(\max_{1 \leq i \leq t} |x_i| + \log t)$.*

The following theorem allows us to rule out the existence of a polynomial kernel for a parameterized problem.

Theorem 4 ([3]). *If an NP-hard problem $L \subseteq \Sigma^*$ has a cross-composition into the parameterized problem Q and Q has a polynomial kernel then coNP $\subseteq$ NP/poly.*

Definition 4 (Turing kernelization). *Let Q be a parameterized problem and let $f : \mathbb{N} \to \mathbb{N}$ be a computable function. A Turing kernelization for Q of size f is an algorithm that decides whether a given instance (x, k) is contained in Q in time polynomial in $|x| + k$, when given access to an oracle that decides membership in Q for any instance (x', k') with $|x'|, k' \leq f(k)$ in a single step. We call such an oracle as f-oracle for Q.*

3 Kernelization Complexity for the Standard Parameter

As noted earlier, the hardness of kernelization for LELP follows by a standard disjoint union argument (see [4] for a similar example). We now demonstrate a cross-composition algorithm for LELP(k) on the class of connected graphs.

We introduce an auxiliary problem, where we seek to find an edge-linked path of length at least k that starts at a specified start vertex s. We denote this variant by LELP$_s(k)$. We will show that this version is NP-complete by a reduction from LELP(k). We will then demonstrate a cross-composition from LELP$_s(k)$ to LELP(k) where the composed instance will turn out to be connected. We defer the details of all these arguments to the full version of the paper due to lack of space. However, to provide some intuition, we describe briefly the construction that we employ to show the NP-completeness of LELP$_s(k)$.

Let (G, k) be an instance of LELP(k). We denote the vertices of G by $\{v_1, \ldots, v_n\}$. Assume, without loss of generality, that $|V(G)| = 2^h$ for some h (if this is not the case, we can obtain an equivalent instance by adding an appropriate number of isolated vertices and at most doubling the size of the instance). Our construction involves a complete binary tree T of height h, and therefore, n leaves. Let r denote the root of T, and let $\ell_1, \ell_2, \ldots, \ell_n$ denote the leaves of T. We make n copies of G, denoted by $G_1, \ldots, G_n$. We make the vertex v_i in the copy G_i adjacent to the leaf ℓ_i. Denote this graph by H. Our reduced instance is now given by $(H, r, k + 2h + 1)$.

We now turn to the Turing kernel. Our goal will be to demonstrate a Turing kernel with $\mathcal{O}(k^{3.656})$ vertices for LELP(k). To explain Turing kernelization for the problem we first define a closely related problem, namely: Long Edge-Linked Cycle. If the first edge and the last edge in an edge-linked path $P = e_1, e_2, \ldots, e_k$ has a common vertex, then we call P an *edge-linked cycle*. We use LELC to refer to the problem of determining if a graph has an edge-linked cycle of length at least k, parameterized by k.

Lemma 1. *If there is a Turing kernel for* LELC *of size f, then there is a Turing kernel for* LELP(k) *of size f for any computable function f.*

Proof Sketch. It is easy to see that if G has an edge-linked path of length k, then there exist $u, v \in V(G)$ such that $G + e$ has an edge-linked cycle of length $k + 1$, where $e = \{u, v\}$. Moreover, for any $u, v \in V(G)$, if $G + e'$ (where $e' = \{u, v\}$) has an edge-linked cycle of length at least $k + 1$, then G has an edge-linked path of length k. As a result to test whether a graph G has an edge linked-path of length k, it is enough to test whether $G + \{\{u, v\}\}$ has an edge-linked cycle of length at least $k + 1$ for some $u, v \in V(G)$. But there is caveat here; we require an oracle for the same problem. In fact it is not hard to show that an oracle for LELC can be obtained using an oracle for LELP(k) and vice versa, where the query lengths are asymptotically the same. □

Thus, in this section we focus on proving the following theorem, which when combined with the hardness result mentioned above, amounts to a proof of Theorem 2.

Theorem 5. *There is a Turing kernel for* LELC *with $\mathcal{O}(k^{3.656})$ vertices and $O(k^{4.656})$ edges.*

We remark that our overall methodology is inspired by the approaches used by [10] to obtain Turing kernels for long path and cycle problems on special

classes of graphs. Towards the proof of Theorem 5 we prove Lemma 2. Before stating Lemma 2 we define the notion of an irrelevant vertex.

Definition 5. *Let (G, k) be an instance of* LELC. *A vertex $v \in V(G)$ is called an* irrelevant *vertex for (G, k) if the following holds: (G, k) is a Yes-instance of* LELC *if and only if $(G - v, k)$ is a Yes-instance of* LELC.

We would like to mention that the size bound of the f-oracle for LELC refers to the bound on the number of vertices in the graph. Moreover, if there is a vertex of degree at least k, then the input is a Yes-instance and thus the bound on the edges will be at most k times the number of vertices.

Lemma 2. *There is a constant c and a function $f : \mathbb{N} \to \mathbb{N}$ defined as $f(x) = cx^{3.656}$ for any $x \in \mathbb{N}$, such that there is a polynomial time algorithm that given an instance (G, k) of* LELC, *uses an f-oracle for* LELC *to:*

- *either correctly determine if (G, k) is a Yes-instance of* LELC, *or,*
- *output an irrelevant vertex $v \in V(G)$.*

Given Lemma 2, the proof of Theorem 5 is straight forward: we can either solve the instance or make progress by deleting the irrelevant vertex. The rest of the section is devoted to prove Lemma 2. To this end, we first introduce and prove some auxiliary lemmas.

Proposition 3. *Let $k \in \mathbb{N}$. Any 3-edge-connected graph (or 3-vertex-connected graph) G with at least $k^{1.328}$ vertices contains an edge-linked cycle of length $\geq k$.*

Proof. Let G be a 3-edge-connected graph, $n = |V(G)|$ and $m = |E(G)|$. Theorem 3 implies the following argument. If $k \leq m^{0.753}$, then G has an Eulerian subgraph with at least k edges, and hence G has an edge-linked cycle of length at least k. Any 3-edge-connected graph G has at least $\frac{3}{2}|V(G)|$ edges. Therefore if $k \leq \left(\frac{3}{2} \cdot n\right)^{0.753}$, then (G, k) is a Yes-instance of LELC. Otherwise $n \leq \frac{2}{3} \cdot k^{1.328}$. Any 3-vertex-connected graph is also a 3-edge-connected graph. □

Lemma 3 ($\star$[3]). *There is a constant c' and a polynomial time algorithm which given an instance (G, k) of* LELC *such that G is connected and $|V(G)| > c'k^{3.656}$, either correctly concludes that (G, k) is a Yes-instance or outputs a separation (A, B) of G of order at most 2 such that $7k < |A| \leq c' \cdot k^{3.656}$.*

The proof of Lemma 3 uses Proposition 2. Next we define an equivalent characterization for edge-linked paths and cycles in terms of trails and tours. A trail is a walk in which no edges repeats and a tour is a closed walk in which no edges repeats. If a trail (tour) covers all the edges in a graph G, then it is a Eulerian path (Eulerian cycle). As a result we have the following simple observations.

[3] The proofs of results marked with $\star$ are deferred to the full version of the paper.

Observation 2. *Let G be a graph. Let C be a tour in G and P be a trail with endpoints u and v, where $u \neq v$. Then, (i) degree of each vertex in the subgraph $(V(C), E(C))$ is even, and (ii) degree of any vertex $w \in V(P) \setminus \{u, v\}$ in the graph $G' = (V(P), E(P))$ is even and degrees of u and v in G' are odd.*

Observation 3. *Let G be a graph. Let C be a connected subgraph of G such that $d_C(v)$ is even for all $v \in V(C)$. Let P be a connected subgraph of G such that there exist two distinct vertices $x, y \in V(P)$ with the following properties: $d_P(x)$ and $d_P(y)$ are odd numbers, and $d_P(v)$ is an even number for all $v \in V(P) \setminus \{x, y\}$. Then, C is a tour, and P is a trail from x to y in G.*

The following observation characterizes edge linked paths and cycles in terms of trails and tours, respectively.

Observation 4. *Let G be a graph. There is an edge linked path of length k in G if and only if there is a trail P in G such that $|E_{in}(V(P))| \geq k$. Also, there is an edge linked cycle of length at least k in G if and only if there is a tour C in G such that $|E_{in}(V(C))| \geq k$.*

Proof. Let $P = u_1 \dots u_\ell$ be a trail in G such that $|E_{in}(V(P))| \geq k$. Let $e_i = \{u_i, u_{i+1}\}$ for all $i \in \{1, \dots, \ell - 1\}$. Let $\{i_1, \dots, i_{\ell'}\} \subseteq \{1, \dots, \ell\}$ be the maximal subset such that u_{i_j} appear first in i_j th position in P for all $j \in \{1, \dots, \ell'\}$. For each $j \in \{1, \dots, \ell'\}$ let S_{i_j} be the set of edges incident to u_{i_j}, but not in $E(P)$ and not incident to any u_{i_r} for some $r < j$. For all $q \notin \{i_1, \dots, i_{\ell'}\}$, $S_q = \emptyset$. Then the sequence of edges in $S_1 e_1 S_2 \dots S_{\ell-1} e_{\ell-1} S_\ell$ forms an edge-linked path containing all the edges in $E_{in}(V(P))$. Similarly, we can prove that there is an edge-linked cycle containing all the edges in $E_{in}(V(C))$. □

Due to Observation 4, to solve LELC, it is enough to test for the existence of a tour C such that $|E_{in}(V(C))| \geq k$. In the following two lemmas we summarize the behaviour of a tour C across a separation of order at most 2.

Lemma 4. *Let G be a graph and (A, B) be a separation of G. Let C be a tour in G such that $|V(C) \cap (A \cap B)| = 1$. Let $x \in V(C) \cap (A \cap B)$. Then, one of the following is true.*

- *C is contained in $G[A]$ or $G[B]$.*
- *$E(C) \cap E(G[A])$ forms a tour C' in $G[A]$ and $x \in V(C')$.*

Proof. Suppose C is contained in $G[A]$ or $G[B]$, then we are done. Otherwise, let F be the subgraph induced on $E(C) \cap E(G[A])$. Clearly $x \in V(F)$. By Observation 2, for any $u \in V(F) \setminus \{x\}$, we know that $d_F(u)$ is even. Then by Observation 1, $d_F(x)$ is even. Since C is connected, F is also connected. Therefore, by Observation 3, F is a tour in $G[A]$. This completes the proof of the lemma. □

Lemma 5 (⋆). *Let G be a graph and (A, B) is a separation of G of order 2. Let C be a tour in G and $A \cap B = \{x, y\} \subseteq V(C)$. Let $G_A = G[A]$ and $G_B = (B, E(G) \setminus E(G_A))$. Then, one of the following is true.*

1. *C is contained in G_B.*
2. *There exists a tour C' in G_A such that $E(C') = E(C) \cap E(G_A)$.*
3. *There exist two vertex disjoint tours C_1 and C_2 in G_A such that $x \in V(C_1), y \in V(C_2)$, and $E(C_1) \cup E(C_2) = E(C) \cap E(G_A)$.*
4. *There exists a trail P from x to y in G_A such that $E(P) = E(C) \cap E(G_A)$.*

By Observation 4, we know that to test for the existence of an edge-linked cycle of length at least k it is enough to test for the existence of a tour such that at least k edges are incident on the vertices of the tour.

Lemma 6. *There exists two constant c_1 and c_2 (where $c_2 < c_1$ and c_2 is same as the constant in Lemma 3), and a function $f : \mathbb{N} \to \mathbb{N}$ defined as $f(x) = c_1 x^{3.656}$ for any $x \in \mathbb{N}$ such that the following holds. There is a polynomial time algorithm $\mathcal{A}$ which given an instance (G, k) of* LELC *and a separation (A, B) of G of order at most 2 such that $7 \cdot k < |A| \leq c_2 \cdot k^{3.656}$ and max degree of G at most k, uses an f-oracle for* LELC *and outputs one of the following.*

- *Correctly concludes that (G, k) is a Yes-instance of* LELC.
- *Outputs an irrelevant vertex $v \in V(G)$.*

Proof. Let $G_A = G[A]$, $G_B = (B, E(G) \setminus E(G_A))$. Algorithm $\mathcal{A}$ first queries the f-oracle with input (G_A, k). If the oracle returns Yes, then $\mathcal{A}$ concludes that (G, k) is a Yes-instance of LELC. From now on we assume that (G_A, k) is a No-instance. We have the following three cases based on $|A \cap B|$.

Case 1 : $|A \cap B| = 0$. In this case any edge-linked cycle is either contained in G_A or contained in G_B. Since (G_A, k) is a No-instance, any vertex in A is an irrelevant vertex.

Case 2 : $|A \cap B| = 1$. Let $\{x\} = A \cap B$. First we prove the following claim.

Claim 1 ($\star$). Let (G_A, k) be a No-instance of LELC. Let $k' < k$ be the largest integer such that there is a tour Q in G_A such that $x \in V(Q)$ and $|E_{in}^{G_A}(V(Q))| = k'$. If there is a tour Q' in $G_A - v$ such that $x \in V(Q')$ and $|E_{in}^{G_A - v}(V(Q'))| = k'$ for some $v \in A \setminus N[x]$, then v is an irrelevant vertex.

We note that the integer k' can be identified using f-oracle as follows. We construct a new graph G' by adding a new cycle D of length $2k$ with vertices x and $2k - 2$ new vertices. Since (G_A, k) is a No-instance, any tour C with $|E_{G'}^{in}(C)| \geq 2k + k'$ contains D (i.e., $E(D) \subseteq E(C)$). Therefore, we can identify k' by calling f-oracle on $(G', 2k + k_1)$ for all $k_1 < k$. The largest k_1 for which $(G', 2k + k_1)$ is a Yes-instance is k'. We will set the value of c_1 to be at least $c_2 + 2$ so that we can use f-oracle. After identifying the value k' we again use f-oracle to identify an irrelevant vertex v. Towards that we query f-oracle with $(G' - u, k' + 2k)$ for all $u \in A \setminus N[x]$ and there will be at least one vertex v for which f-oracle outputs Yes on input $(G' - v, k' + 2k)$. That vertex v is an irrelevant vertex because of Claim 1.

Case 3 : $|A \cap B| = 2$. Let $A \cap B = \{x, y\}$. First we prove the following claim.

Claim 2. Suppose there is a path between x and y in G_B and there is trail P from x to y in G_A such that $|E_{in}^{G_A}(V(P))| \geq k-1$. Then (G, k) is a Yes instance.

Proof. Let P' be a path from x to y in G_B. Then the edges of P and P' forms a tour C in G. Since $E(P')$ is disjoint from $E(G_A)$ and $|E_{in}^{G_A}(V(P))| \geq k-1$, we have that $|E_{in}^{G}(V(C))| \geq k$. Then, by Observation 4, (G, k) is a Yes-instance. □

Algorithm $\mathcal{A}$ will test whether there is a path between x and y in G_B and the existence of a trail P from x to y in G_A with $|E_{in}^{G_A}(V(P))| \geq k-1$ using f-oracle as follows. We construct a graph G' by adding a $2k$ length path L between x and y. Since (G_A, k) is a No-instance, and any tour C with $|E_{in}^{G'}(V(P))| \geq 3k-1$ will contain the path L. Therefore $(G', 3k-1)$ is a Yes-instance of LELC if and only if there is a trail P from x to y in G_A such that $|E_{in}^{G_A}(V(P))| \geq k-1$. Then by Claim 2, we conclude that (G, k) is a Yes instance.

Now on we assume that if G_B contains a path between x and y, then there is no trail P from x to y in G_A with $|E_{in}^{G_A}(V(P))| \geq k-1$. We use f-oracle to output an irrelevant vertex as follows. We construct three graphs G_1, G_2, G_3, and G_4 as follows. The graph G_1 is created by adding a cycle L_1 of length $4k$ containing x and $4k-1$ new vertices. The graph G_2 is created by adding a cycle L_2 of length $4k$ containing y and $4k-1$ new vertices. The graph G_3 is created by adding two internally vertex disjoint paths Q_3 and Q_3' of length $2k$ each between x and y. The graph G_4 is created by adding a path Q_4 of length $4k$ between x and y. For each $i \in \{1, \ldots, 4\}$, we use f-oracle to compute the largest $1 \leq k_i < k$ such that $(G_i, 4k + k_i)$ is a Yes-instance. If no such k_i exists we set $k_i = -\infty$. Clearly this can be done by querying the f-oracle at most k times for each $i \in \{1, \ldots, 4\}$. For any $i \in \{1, \ldots, 4\}$, if $k_i \neq -\infty$, any tour C_i in G_i with $|E_{in}^{G_i}(V(C_i))| \geq 4k + k_i$ will contain the newly added $4k$ edges. Therefore all but k_i vertices in G_A are irrelevant for the instance $(G_i, 4k + k_i)$. These vertices can be identified using f-oracle. So Algorithm $\mathcal{A}$ identifies at most $4k$ vertices that are relevant for at least one of $(G_i, 4k + k_i)$. Since $|A| > 7k$, there is one vertex $v \in A \setminus (N[x] \cup N[y])$ which is irrelevant for $(G_i, 4k + k_i)$ for all $i \in \{1, \ldots, 4\}$. Algorithm $\mathcal{A}$ will output such a vertex v as an irrelevant vertex for the instance (G, k). One can show that vertex v is indeed an irrelevant vertex for (G, k). This concludes the proof of the Lemma.

□

We are now ready to give the proof of Lemma 2.

Proof of Lemma 2. First we set constants c_1 and c_2 mentioned by Lemma 6. Then we set the constant c mentioned in the lemma to be c_1. Let (G, k) be the input instance. If there is a vertex $v \in V(G)$ with degree at least k, then we conclude that (G, k) is a Yes-instance. Let H be a connected component of G. If $|V(H)| \leq c_2 k^{3.656}$, then we use f-oracle to test whether (H, k) is a Yes-instance or not. If (H, k) is not a Yes-instance, then we output any vertex in $V(H)$ as an irrelevant vertex.

Now consider the case $|V(H)| > c_2 k^{3.656}$. We use Lemma 3 on (H, k). If Lemma 3 concludes that (H, k) is a Yes-instance, then (G, k) is indeed a Yes-instance. Otherwise Lemma 3 outputs a separation (A, B') of H of order at most 2 such that $7k < |A| \leq c_2 \cdot k^{3.656}$. Notice that $(A, B = B' \cup (V(G) \setminus V(H)))$ is a separation of G of order at most 2. Now we use Lemma 6 to either conclude that (G, k) is a Yes-instance or to output an irrelevant vertex. □

4 Concluding Remarks

In this contribution, we introduced and studied a natural optimization variant of the notion of an edge Hamiltonian path. Specifically, we ask if a graph admits such an edge-linked path of length at least k. We show that the problem is FPT when parameterized by k, and admits a single-exponential algorithm when parameterized by treewidth. The latter generalizes and improves known algorithms for the special case of the Edge Hamiltonicity problem (where $k = m$) when parameterized by treewidth. While we used the technique of representative families, possibly, other techniques can also be employed to obtain similar running time dependency on the treewidth parameter (for example, using rank-based approaches [5]).

We also studied the kernelization complexity of the problem, showing that it is unlikely to admit a polynomial kernel even if the input graph is connected. Our main result in this context was a Turing kernel for the problem, which made use of Tutte decompositions and the existence of long edge-linked paths on 3-connected graphs. An interesting open problem here is to determine the kernelization complexity of the problem on two-connected graphs. We conjecture that the problem remains hard for this class of graphs as well.

References

1. Bertossi, A.A.: The edge Hamiltonian path problem is NP-complete. Inf. Process. Lett. **13**(4/5), 157–159 (1981)
2. Bilinski, M., Jackson, B., Ma, J., Yu, X.: Circumference of 3-connected claw-free graphs and large Eulerian subgraphs of 3-edge-connected graphs. J. Comb. Theory Ser. B **101**(4), 214–236 (2011)
3. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. SIAM J. Discrete Math. **28**(1), 277–305 (2014)
4. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
5. Cygan, M., Kratsch, S., Nederlof, J.: Fast hamiltonicity checking via bases of perfect matchings. J. ACM **65**(3), 12:1–12:46 (2018)
6. Demaine, E.D., Demaine, M.L., Harvey, N.J.A., Uehara, R., Uno, T., Uno, Y.: UNO is hard, even for a single player. Theor. Comput. Sci. **521**, 51–61 (2014)
7. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173, 4th edn. Springer, Heidelberg (2012)
8. Hopcroft, J., Tarjan, R.: Dividing a graph into triconnected components. SIAM J. Comput. **2**(3), 135–158 (1973)

9. Jansen, B.M.P., Pilipczuk, M., Wrochna, M.: Turing kernelization for finding long paths in graphs excluding a topological minor. In: IPEC, vol. 89. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
10. Jansen, B.M.: Turing kernelization for finding long paths and cycles in restricted graph classes. J. Comput. Syst. Sci. **85**(C), 18–37 (2017)
11. Lai, T.H., Wei, S.S.: The edge Hamiltonian path problem is NP-complete for bipartite graphs. Inf. Process. Lett. **46**(1), 21–26 (1993)
12. Lampis, M., Makino, K., Mitsou, V., Uno, Y.: Parameterized edge hamiltonicity. Discrete Appl. Math. **248**, 68–78 (2018)
13. Ryjáček, Z., Woeginger, G.J., Xiong, L.: Hamiltonian index is NP-complete. Discrete Appl. Math. **159**(4), 246–250 (2011)
14. Thomassé, S., Trotignon, N., Vuskovic, K.: A polynomial turing-kernel for weighted independent set in bull-free graphs. Algorithmica **77**(3), 619–641 (2017)