Electronic Journal of Graph Theory and Applications **7** (2) (2019), 411–428

Electronic Journal of
Graph Theory and Applications

Cycle decompositions and constructive characterizations

Irene Heinrich, Manuel Streicher

Optimization Research Group,
Faculty of Mathematics, Technische Universität Kaiserslautern,
Paul-Ehrlich Straße 14, 67663 Kaiserslautern

irene.heinrich10.10.15@gmail.com, streicher@mathematik.uni-kl.de

Abstract

Decomposing an Eulerian graph into a minimum respectively maximum number of edge disjoint cycles is an NP-complete problem. We prove that an Eulerian graph decomposes into a unique number of cycles if and only if it does not contain two edge disjoint cycles sharing three or more vertices. To this end, we discuss the interplay of three binary graph operators leading to novel constructive characterizations of two subclasses of Eulerian graphs. This enables us to present a polynomial-time algorithm which decides whether the number of cycles in a cycle decomposition of a given Eulerian graph is unique.

Keywords: constructive characterization, cycle decomposition, cycle packing, cycle covering, Eulerian graphs
Mathematics Subject Classification : 05C38, 05C40, 05C45, 05C70
DOI: 10.5614/ejgta.2019.7.2.15

1. Introduction

It is well-known that a graph is Eulerian if and only if its edge set can be decomposed into cycles (cf. [3]). The decision problem whether an Eulerian graph can be decomposed into at most k cycles is NP-complete as a consequence of [10]. Also the corresponding maximization problem is NP-complete, cf. [4]. Approximation algorithms for the maximization problem are discussed in [6] and [9].

Received: 23 November 2018, Revised: 30 May 2019, Accepted: 15 June 2019.

Our contribution is to give two equivalent characterizations for the class of Eulerian graphs where both numbers – the minimum and the maximum amount of cycles in a cycle decomposition – coincide. We show that those are exactly the graphs that can be constructed from the set of Eulerian multiedges using a finite number of vertex-identifications and vertex-edge-identifications which will be introduced and discussed in Section 3. This constructive characterization then enables us to prove the following statement:

Theorem (5.2 - *shortened version*)**.** *Let G be an Eulerian graph. The number of cycles in a cycle decomposition of G is unique if and only if no two edge-disjoint cycles in G intersect more than twice.*

We exploit Theorem 5.2 to develop an algorithm which applies the identification operations backwards. We can recognize the described graph class in polynomial time.

Theorem (6.6 - *shortened version*)**.** *We can decide in time $\mathcal{O}(n(n+m))$ if the number of cycles in a cycle decomposition of a given Eulerian graph is unique.*

Our main tool for proving the before mentioned results is a novel *constructive characterization*. A constructive characterization of a graph class is a construction manual for building all graphs in the class starting from some simple set of initial graphs. Many graph classes can be expressed through constructive characterization, among those are graphs of low treewidth [1], 3-connected [11] and k-edge-connected graphs [8].

We may turn the before mentioned statement – *a graph is Eulerian if and only if it is connected and can be decomposed into cycles* – into a toy example for a constructive characterization. We describe the class $\mathcal{E}$ of Eulerian graphs recursively:

- If G is isomorphic to K_1 or C_n for some $n \in \mathbb{N}, n \geq 1$, then $G \in \mathcal{E}$.
- If $G_1, G_2 \in \mathcal{E}$ with $E(G_1) \cap E(G_2) = \emptyset$ and $V(G_1) \cap V(G_2) \neq \emptyset$, then also $G_1 \cup G_2 \in \mathcal{E}$.

Often constructive characterizations can be exploited to prove a desired statement by induction. Coming back to the above toy example, we can prove that every Eulerian graph has only vertices of even degree by first observing that each C_n and the K_1 have only vertices of even degree and then using the fact that the graph union with disjoint edge sets does not change the even degrees.

We study three basic binary graph operators. In Section 3 we define these operators and regard their behaviour concerning the following graph invariants: connectivity, minimum and maximum number of cycles in a cycle decomposition and treewidth. Sections 4 and 5 will then use the introduced operators for constructive characterizations of Eulerian graphs with maximum degree at most 4 and treewidth at most 2 (Section 4) and Eulerian graphs which have the property that the number of cycles in all of its cycle decompositions is the same (Section 5). Finally, in Section 6 we exploit the gained insights to develop a polynomial time algorithm which decides if the cycle number of a given Eulerian graph is unique.

2. Preliminaries

We use standard graph terminology, see [12, 2, 7]. Though, we recall some basic notions in the following. A *graph* G is a triple consisting of a finite non-empty *vertex set* $V(G)$, and a finite *edge set* $E(G)$ and a relation that associates with each edge two different vertices called its *end vertices*. Observe that this definition excludes loop edges. If two edges have the same two endvertices we call them *parallel*. An edge with end vertices u and v is often written as uv. We use this notation even if G has parallel edges between the vertices u and v. This does not lead to any inconvenience as the problems discussed in this article refer to graph invariants which do not depend on the choice of the exact edge between u and v. We denote with $N_G(u)$ the set of all neighbours of u in G. The *degree* $\deg_G(v)$ of $v \in V(G)$ is defined as the number of edges incident to v. If all vertices of G have the same degree k, then G is k-regular. We call two graphs G and G' disjoint if $V(G) \cap V(G') = \emptyset$ and $E(G) \cap E(G') = \emptyset$. Let G and G' be two graphs with $E(G) \cap E(G') = \emptyset$. We set $G \cup G'$ to be the graph with $V(G \cup G') = V(G) \cup V(G')$ and $E(G \cup G') = E(G) \cup E(G')$.

Let $u \in V(G)$. We denote with $G - u$ the graph where u and all its incident edges are removed from G. For $F \subseteq E(G)$ we write $G - F$ for the graph with $V(G - F) = V(G)$ and $E(G - F) = E(G) \setminus F$. If $F = \{f\}$ we write $G - f$. Let G be a graph containing a vertex $u \in V(G)$ with $\deg_G(u) = 2$ with two distinct neighbours. *Resolving* u means to remove u from G and to connect its two neighbours by a new edge. A *path* P is a graph of the form $V(P) = \{u_0, u_1, \ldots, u_k\}$, $E(P) = \{u_0u_1, u_1u_2, \ldots, u_{k-1}u_k\}$, where all the u_i are distinct. We often refer to a path omitting its precise edges but only listing the sequence of its vertices ordered according to their appearance in P, say $P = u_0u_1 \ldots u_k$. We say that P is a u_0-u_k-path, the vertices $u_1, \ldots, u_{k-1}$ are *internal* vertices of P. Let P be a u-v-path and Q be a v-w-path with $V(P) \cap V(Q) = v$. We set $PQ \coloneqq P \cup Q$. Even when we study paths as subgraphs of non-simple graphs, this notation does not lead to any inconvenience: In the upcoming topics it is never of any relevance which precise edge a path uses. If $P = u_0 \ldots u_k$ is a path, then the graph $C \coloneqq P \cup u_ku_0$ is a *cycle*. A *cycle decomposition* of a graph G is a set of cycles which are subgraphs in G such that each edge appears in exactly one cycle in the set. We set

$$c(G) \coloneqq \min\{|\mathcal{C}| \colon \mathcal{C} \text{ is a cycle decomposition of } G\},$$
$$\nu(G) \coloneqq \max\{|\mathcal{C}| \colon \mathcal{C} \text{ is a cycle decomposition of } G\}$$

to be the *minimum* respectively *maximum cycle number* of G. A graph is *Eulerian* if it allows for an *Euler tour*, i.e. a non-empty alternating sequence $v_0e_0v_1e_1 \ldots e_{k-1}v_k$ of vertices and edges in G such that e_i has end vertices v_i and v_{i+1} for all $0 \leq i < k$, $v_0 = v_k$ and every edge of G appears exactly once in the sequence.

A graph G is called *connected* if it is non-empty and any two of its vertices are linked by a path in G. The *components* of a graph are its maximal (with respect to the subgraph relation) connected subgraphs. For $V_1, V_2 \subseteq V(G)$ we set $E(V_1, V_2)$ to be the set of all edges with one endvertex in V_1 and the other endvertex in V_2. A set F of edges is a *cut* in G if there exists a partition $\{V_1, V_2\}$ of V such that $F = E(V_1, V_2)$. We call F a k-cut if $|F| = k$. An element of a 1-cut is called a *cut-edge*. A connected graph G is called k-edge-connected if it stays connected after the removal of $k - 1$ arbitrary edges. A vertex $v \in V(G)$ is a *cut-vertex* if $G - v$ has more connected components than G. A connected graph without cut-vertices is called *biconnected*. The maximal biconnected subgraphs of a graph are called its *biconnected components*. For a more detailed description of biconnectivity and some basic results we refer to [12] and [5]. We say that a set $S \subseteq V(G) \cup E(G)$ *separates* $w_1, w_2 \in V(G)$ if there exists no w_1-w_2-path in G without elements of S.

A connected graph T that does not contain a cycle as a subgraph is a *tree*. A vertex of degree 1 in T is called a leaf. For a graph G a *tree-decomposition* $(\mathcal{T}, \mathcal{B})$ of G consists of a tree $\mathcal{T}$ and a set $\mathcal{B} = \{B_t \colon t \in V(\mathcal{T})\}$ of *bags* $B_t \subseteq V(G)$ such that $V(G) = \bigcup_{t \in V(\mathcal{T})} B_t$, for each edge $vw \in E(G)$ there exists a vertex $t \in V(\mathcal{T})$ such that $v, w \in B_t$, and if $v \in B_s \cap B_t$, then $v \in B_r$ for each vertex r on the path connecting s and t in $\mathcal{T}$. A tree-decomposition $(\mathcal{T}, \mathcal{B})$ has *width* k if each bag has a size of at most $k+1$ and there exists some bag of size $k+1$. The *treewidth* of G is the smallest integer k for which there is a width k tree-decomposition of G. We write $\mathrm{tw}(G) = k$. A tree-decomposition $(T, \mathcal{B})$ of width k is *smooth* if $|B_t| = k+1$ for all $t \in V(\mathcal{T})$ and $|B_s \cap B_t| = k$ for all $st \in E(\mathcal{T})$. A graph of treewidth at most k always has a smooth tree-decomposition of width k; see Bodlaender [1].

The *contraction* of an edge e with endpoints u, v is the replacement of u and v with a single vertex whose incident edges are the edges other than e that were incident to u or v. A graph H is a *minor* of a graph G if an isomorphic copy of H can by obtained from G by deleting or contracting edges of G. The graph H obtained by *subdivision* of some edge $uv \in E(G)$ is obtained by replacing the edge uv by a new vertex w and edges uw and wv.

3. Construction operations

In the following, we introduce three binary graph operations – vertex-identification, edge-identification and vertex-edge-identification. The constructive characterizations in Section 4 and 5 each start off by a simple base class of graphs. In Section 4 the considered class is then built by mainly using edge-identification. In Section 5 the vertex-edge-identification is the crucial construction tool. After defining the above mentioned operations we regard their behaviour concerning cycle decompositions, connectivity and treewidth.

Vertex-identification. Let G_1, G_2 be disjoint graphs and let $u_1 \in V(G_1), u_2 \in V(G_2)$. We construct the graph $(G_1, u_1) \bullet (G_2, u_2)$ by identifying u_1 and u_2.

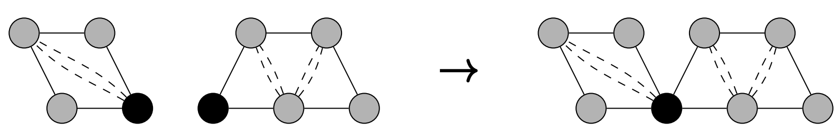

Figure 1. Vertex-identification of two Eulerian graphs.

Edge-identification. Let G_1, G_2 be disjoint graphs. Further let $e_i \in E(G_i)$ be an edge with endpoints u_i, v_i for $i \in \{1, 2\}$. We construct the graph $(G_1, e_1, u_1) = (G_2, e_2, u_2)$ by removing the edge e_i from G_i, $i \in \{1, 2\}$ and adding an edge from u_1 to u_2 and another one from v_1 to v_2.

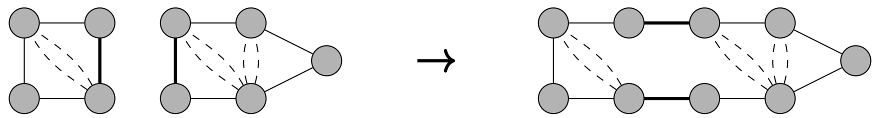

Figure 2. Edge-identification of two Eulerian graphs.

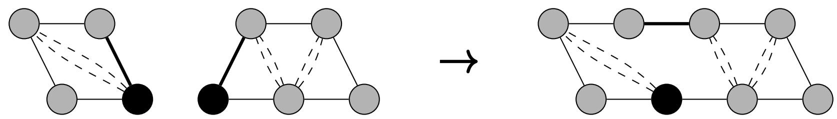

Figure 3. Vertex-edge-identification of two Eulerian graphs.

Vertex-edge-identification. Let G_1, G_2 be disjoint graphs and let e_i be an edge in G_i from u_i to v_i for $i \in \{1,2\}$. We define $(G_1, e_1, u_1) \mathbin{\bar{\bullet}} (G_2, e_2, u_2)$ to be the graph where v_1 and v_2 are identified, the edges e_1, e_2 are removed and an edge between u_1 and u_2 is added.

If e_i and u_i are clear from the context or the statement is independent from the choice of e_i and u_i then we simply write $G_1 \bullet G_2$, $G_1 \mathbin{\boldsymbol{=}} G_2$ and $G_1 \mathbin{\bar{\bullet}} G_2$.

Cycles invariants are compatible with the identification operations. In the following we demonstrate that the identification operations preserve the cycle behaviour in a natural way. We just keep all cycles whose edges are untouched by the operation (in the case of vertex identification these are all cycles). In each of G_1 and G_2 exactly one edge is deleted in the construction of $G_1 \mathbin{\boldsymbol{=}} G_2$ respectively $G_1 \mathbin{\bar{\bullet}} G_2$. We obtain a cycle in $G_1 \mathbin{\boldsymbol{=}} G_2$ respectively $G_1 \mathbin{\bar{\bullet}} G_2$ which uses the edges not contained in $E(G_1)$ nor in $E(G_2)$ by combining a cycle from G_1 with a cycle from G_2 each containing a deleted edge.

Lemma 3.1 (Cycle invariants under construction operations)**.** *Let G_1 and G_2 be Eulerian graphs.*

(i) The invariants c and ν show the following behaviour under vertex-identification:

$$c(G_1 \bullet G_2) = c(G_1) + c(G_2),$$
$$\nu(G_1 \bullet G_2) = \nu(G_1) + \nu(G_2).$$

(ii) They behave in the following way under edge-identification and vertex-edge-identification:

$$c(G_1 \mathbin{\boldsymbol{=}} G_2) = c(G_1 \mathbin{\bar{\bullet}} G_2) = c(G_1) + c(G_2) - 1,$$
$$\nu(G_1 \mathbin{\boldsymbol{=}} G_2) = \nu(G_1 \mathbin{\bar{\bullet}} G_2) = \nu(G_1) + \nu(G_2) - 1.$$

Proof.

(i) For $i \in \{1,2\}$ let $v_i \in V(G_i)$ such that $G_1 \bullet G_2 = (G_1, v_1) \bullet (G_2, v_2)$. The vertex which arises from the identification of $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$ is a cut-vertex. Thus, we obtain a one-to-one-correspondence of cycle decompositions in $G_1 \cup G_2$ and cycle decompositions in $G_1 \bullet G_2$ just by relabelling v_1 and v_2 to v and adjusting the incident edges. Altogether we obtain $c(G_1 \bullet G_2) = c(G_1) + c(G_2)$ and $\nu(G_1 \bullet G_2) = \nu(G_1) + \nu(G_2)$.

(ii) Let $G_1 \mathbin{\boldsymbol{=}} G_2 = (G_1, e_1, u_1) \mathbin{\boldsymbol{=}} (G_2, e_2, u_2)$ for suitable $e_i \in E(G_i)$ and $u_i \in V(G_i)$, $i \in \{1, 2\}$. In a cycle decomposition of G_i there is exactly one cycle C_i containing the edge e_i with end vertices u_i and v_i. We obtain a one-to-one-correspondence of cycle decompositions in $G_1 \cup G_2$ and cycle decompositions in $G_1 \mathbin{\boldsymbol{=}} G_2$ by keeping all cycles from the decompositions of G_1 and G_2 except C_1 and C_2 and adding the cycle C with $E(C) = (E(C_1) \cup E(C_2) \setminus \{u_1v_1, u_2v_2\}) \cup \{u_1u_2, v_1v_2\}$, see Figure 2. Thus,

$$c(G_1 \mathbin{\boldsymbol{=}} G_2) = c(G_1 \cup G_2) - 1 = c(G_1) + c(G_2) - 1 \text{ and}$$
$$\nu(G_1 \mathbin{\boldsymbol{=}} G_2) = \nu(G_1 \cup G_2) - 1 = \nu(G_1) + \nu(G_2) - 1.$$

Now let $G_1 \mathbin{\bar{\bullet}} G_2 = (G_1, e_1, u_1) \mathbin{\bar{\bullet}} (G_2, e_2, u_2)$ for suitable $e_i \in E(G_i)$, $u_i \in V(G_i)$, $i \in \{1, 2\}$. Analogously to the previous operation, we obtain a one-to-one correspondence between cycle decompositions of $G_1 \cup G_2$ and $G_1 \mathbin{\bar{\bullet}} G_2$ by choosing C with $E(C) = (E(C_1) \cup E(C_2) \cup \{u_1u_2\}) \setminus \{e_1, e_2\}$, see Figure 3. Consequently we obtain the same relations as above: $c(G_1 \mathbin{\bar{\bullet}} G_2) = c(G_1) + c(G_2) - 1$ and $\nu(G_1 \mathbin{\bar{\bullet}} G_2) = \nu(G_1) + \nu(G_2) - 1$. □

Corollary 3.2. *Let G_1, G_2 be two Eulerian graphs. If $G = G_1 \circ G_2$ for some $\circ \in \{\bullet, \boldsymbol{=}, \bar{\bullet}\}$ then it holds true that*

$$\nu(G) - c(G) = (\nu(G_1) - c(G_1)) + (\nu(G_2) - c(G_2)).$$

Connectivity is compatible with the identification operations. We show in Lemma 3.4 that the behaviour of paths between two given vertices in G_1 is preserved in $G_1 \mathbin{\bar{\bullet}} G_2$. We follow the intuition to keep all paths which do not contain the edge of G_1 which is deleted in $G_1 \mathbin{\bar{\bullet}} G_2$ and to reroute a path which uses the deleted edge along a path in G_2. We translate the results to the construction $G_1 \mathbin{\boldsymbol{=}} G_2$. We start off by the observation that cut-edges are preserved under vertex-edge-identification.

Observation 3.3. *Let G_1 be a graph containing a cut-edge and let G_2 be some other graph. Then, also $G_1 \mathbin{\bar{\bullet}} G_2$ contains a cut-edge.*

Proof. Let $e_i \in E(G_i)$ and $u_i \in V(G_i)$ for $i \in \{1, 2\}$ such that $G_1 \mathbin{\bar{\bullet}} G_2 = (G_1, e_1, u_1) \mathbin{\bar{\bullet}} (G_2, e_2, u_2)$. Let $e' \in E(G_1)$ be a cut-edge. If $e' \neq e_1$ then e' is still a cut-edge in $G_1 \mathbin{\bar{\bullet}} G_2$. Otherwise, the new edge connecting u_1 and u_2 is a cut-edge in $G_1 \mathbin{\bar{\bullet}} G_2$. □

Lemma 3.4. *Let G_1 and G_2 be 2-edge-connected graphs with edges $e_i = v_iu_i \in E(G_i)$ for $i \in \{1, 2\}$. Let e be the edge in $(G_1, e_1, u_1) \mathbin{\bar{\bullet}} (G_2, e_2, u_2)$ with end vertices u_1 and u_2 and let v be the vertex arising from the identification of v_1 and v_2. Let $S \subseteq V(G_1) \cup E(G_1)$. Further set*

$$S' := \begin{cases} S, & \text{if } v_1, e_1 \notin S, \\ (S \setminus \{e_1\}) \cup \{e\}, & \text{if } e_1 \in S, v_1 \notin S, \\ (S \setminus \{v_1\}) \cup \{v\}, & \text{if } v_1 \in S, e_1 \notin S, \\ (S \setminus \{e_1, v_1\}) \cup \{e, v\}, & \text{if } v_1, e_1 \in S. \end{cases}$$

Let $w_1, w_2 \in V(G_1)$ be two distinct vertices. We may assume $w_1 \neq v_1$. Set

$$w_2' := \begin{cases} w_2, & \text{if } w_2 \neq v_1, \\ v, & \text{if } w_2 = v_1. \end{cases}$$

The set S separates w_1 and w_2 in G_1 if and only if S' separates w_1 and w_2' in $G_1 \bar{\bullet} G_2$.

Proof. If suffices to show: G_1 contains a w_1-w_2-path without elements from S if and only if $G_1 \bar{\bullet} G_2$ contains a w_1-w_2'-path without elements from S'.

Let P be a w_1-w_2-path in G_1 with $(V(P) \cup E(P)) \cap S = \emptyset$. Assume that P does not contain v_1 and e_1, then $w_2' = w_2$ and P is a w_1-w_2'-path in $G_1 \bar{\bullet} G_2$ with $(V(P) \cup E(P)) \cap S' = \emptyset$. Now assume that P contains v_1 but not e_1. We obtain a w_1-w_2'-path P' with $(V(P) \cup E(P)) \cap S' = \emptyset$ by renaming v_1 to v in P. Last assume that P contains e_1. Then, P is of the form $P = P_1 u_1 e_1 v_1 P_2$ where P_1 is a w_1-u_1-path (resp. w_2-u_1-path) and P_2 is a v_1-w_2-path (resp. v_1-w_1-path) in G_1. From the 2-edge-connectivity of G_2, we obtain that there exists a u_2-v_2-path Q in $G_2 - e_2$. Let Q' be the path obtained from Q by renaming v_2 to v and let P_2' be the path in $G_1 \bar{\bullet} G_2$ obtained from P_2 by renaming v_1 to v. Now $P_1 u_1 e u_2 Q' P_2'$ is a w_1-w_2-path in $G_1 \bar{\bullet} G_2$ with $(V(P_1 Q' P_2') \cup E(P_1 Q' P_2')) \cap S' = \emptyset$.

Let now P' be a w_1-w_2'-path in $G_1 \bar{\bullet} G_2$ with $(V(P') \cup E(P')) \cap S' = \emptyset$. If $V(P') \subseteq V(G_1) \cup \{v\}$ then the path obtained from P' by renaming v to v_1 (if it is contained in P') is a w_1-w_2-path in G_1. Otherwise P' must be of the form $P' = P_1' u_1 e u_2 P_2' P_3'$, where P_1' is a w_1-u_1-path (resp. w_2-u_1-path) with edges in $E(G_1) \setminus \{e_1\}$, P_2' is a u_2-v-path with edges in $E(G_2) \setminus \{e_2\}$ and P_3' is a v-w_2-path (resp. v-w_1-path) with edges in $E(G_1) \setminus \{e_1\}$. Let P_3 be the path in G_1 that arises from P_3' by renaming v to v_1. We obtain $(V(P_1' u_1 e_1 v_1 P_3) \cup E(P_1' u_1 e_1 v_1 P_3)) \cap S = \emptyset$ and $P_1' u_1 e_1 v_1 P_3$ is a w_1-w_2-path in G_1. □

Corollary 3.5. *Let G_1 and G_2 be graphs with edges $e_i = v_i u_i \in E(G_i)$ for $i = 1, 2$. Let e be the edge in $(G_1, e_1, u_1) \bar{\bullet} (G_2, e_2, u_2)$ with end vertices u_1 and u_2 and let v be the vertex arising from the identification of v_1 and v_2. It holds that $G_1 \bar{\bullet} G_2$ is biconnected if and only if G_1 and G_2 are biconnected and contain more than one edge.*

Proof. Assume that G_1 and G_2 are both biconnected and each contain more than one edge. Let $w \in V(G_1)$ and $x \in V(G_2)$. Then, by Menger's Theorem (see [12]) there exist internally vertex disjoint paths P_1 from w to u_1 and Q_1 from w to v_1 in G_1. Further, there exist internally vertex disjoint paths P_2 from u_2 to x and Q_2 from v_2 to x in G_2. Let for $i \in \{1, 2\}$ Q_i' be the path that arises from Q_i by renaming v_i to v. Now $P_1 u_1 e u_2 P_2$ and $Q_1' Q_2'$ are two internally vertex disjoint w-x-paths in $G_1 \bar{\bullet} G_2$. For $i \in \{1, 2\}$ and two vertices w_1 and w_2 in $V(G_i) \setminus \{v_i\}$ we obtain from Lemma 3.4 and Menger's Theorem that there exists two internally vertex disjoint paths in $G_1 \bar{\bullet} G_2$ connecting w_1 and w_2.

If G_i for some $i \in \{1, 2\}$ contains just one edge, then $G_1 \bar{\bullet} G_2$ contains a cut vertex. Next suppose that G_i has a cut-edge for some $i \in \{1, 2\}$. By Observation 3.3 also $G_1 \bar{\bullet} G_2$ has a cut-edge. Last suppose that G_1 and G_2 are 2-edge-connected and G_i has a cut-vertex for some $i \in \{1, 2\}$. But then by Lemma 3.4 also $G_1 \bar{\bullet} G_2$ has a cut-vertex. This settles the claim. □

Lemma 3.6. *Let G_1 and G_2 be 2-edge connected graphs and let $e_i \in E(G_i)$ with end vertices u_i and v_i for $i \in \{1,2\}$. Let $S \subseteq V(G_1) \cup E(G_1)$. Let $w_1, w_2 \in V(G_1)$ be two distinct vertices. Set*

$$S' := \begin{cases} S, & \text{if } e_1 \notin S, \\ (S \setminus \{e_1\}) \cup \{u_1u_2\}, & \text{if } e_1 \in S. \end{cases}$$

Then, S separates w_1 and w_2 in G_1 if and only if S' separates w_1 and w_2 in $(G_1, e_1, u_1) \mathbin{\boldsymbol{=}} (G_2, e_2, u_2)$. In particular, $G_1 \mathbin{\boldsymbol{=}} G_2$ is biconnected if and only if G_1 and G_2 are biconnected.

Proof. The proof is analogous to the proofs of Lemma 3.4 and Corollary 3.5. □

Treewidth is compatible with the identification operations. Also the treewidth behaves nicely with the identification operations. Clearly, the treewidth of a graph can be computed knowing the treewidth of its biconnected components. Furthermore, a width-optimal tree decomposition of $G_1 \mathbin{\boldsymbol{=}} G_2$ or $G_1 \mathbin{\bar{\bullet}} G_2$ can be constructed by just slighlty changing a tree decomposition of $G_1 \cup G_2$. The results are summarized in Lemma 3.7.

Lemma 3.7. *Let G_1 and G_2 be 2-edge-connected graphs. It holds true that*

$$\begin{aligned} \operatorname{tw}(G_1 \bullet G_2) &= \max\{\operatorname{tw}(G_1), \operatorname{tw}(G_2)\}, \\ \operatorname{tw}(G_1 \mathbin{\boldsymbol{=}} G_2) &= \max\{2, \operatorname{tw}(G_1), \operatorname{tw}(G_2)\} \text{ and} \\ \operatorname{tw}(G_1 \mathbin{\bar{\bullet}} G_2) &= \max\{2, \operatorname{tw}(G_1), \operatorname{tw}(G_2)\}. \end{aligned}$$

Proof. A graph is of treewidth at most k if and only if all of its biconnected components are of treewidth at most k, cf. [1]. Thus, $\operatorname{tw}(G_1 \bullet G_2) = \max\{\operatorname{tw}(G_1), \operatorname{tw}(G_2)\}$.

By the assumption that G_1 and G_2 are 2-edge connected we obtain that $G_1 \mathbin{\boldsymbol{=}} G_2$ and $G_1 \mathbin{\bar{\bullet}} G_2$ each contain a cycle of length not less than 3. Consequently $\operatorname{tw}(G_1 \mathbin{\boldsymbol{=}} G_2) \geq 2$ and $\operatorname{tw}(G_1 \mathbin{\bar{\bullet}} G_2) \geq 2$. Now, G_1 and G_2 are minors of $G_1 \mathbin{\boldsymbol{=}} G_2$ and $G_1 \mathbin{\bar{\bullet}} G_2$. Altogether $\operatorname{tw}(G_1 \mathbin{\boldsymbol{=}} G_2) \geq \max\{2, \operatorname{tw}(G_1), \operatorname{tw}(G_2)\}$ and $\operatorname{tw}(G_1 \mathbin{\bar{\bullet}} G_2) \geq \max\{2, \operatorname{tw}(G_1), \operatorname{tw}(G_2)\}$. For the other inequality let $(\mathcal{T}^{(i)}, \mathcal{B}^{(i)})$ be a tree decomposition of G_i and let $B_i \in \mathcal{B}^{(i)}$ be a bag with $\{u_i, v_i\} \in B_i$ for $i \in \{1,2\}$. We obtain a tree decomposition of $G_1 \mathbin{\boldsymbol{=}} G_2$ of width $\max\{2, \operatorname{tw}(G_1), \operatorname{tw}(G_2)\}$ by the following construction. Set

$$\begin{aligned} B_a &:= \{u_1, u_2, v_1\}, \\ B_b &:= \{u_2, v_1, v_2\}, \\ \mathcal{B} &:= \mathcal{B}^{(1)} \cup \mathcal{B}^{(2)} \cup \{B_a, B_b\}, \\ V(\mathcal{T}) &:= V(\mathcal{T}^{(1)}) \cup V(\mathcal{T}^{(2)}) \cup \{a, b\} \text{ and} \\ E(\mathcal{T}) &:= E(\mathcal{T}^{(1)}) \cup E(\mathcal{T}^{(2)})) \cup \{1a, ab, b2\}. \end{aligned}$$

Now $(\mathcal{T}, \mathcal{B})$ is a tree decomposition of $G_1 \mathbin{\boldsymbol{=}} G_2$ of width $\max\{2, \operatorname{tw}(G_1), \operatorname{tw}(G_2)\}$. The inequality for $G_1 \mathbin{\bar{\bullet}} G_2$ follows immediately since $G_1 \mathbin{\bar{\bullet}} G_2$ is a minor of $G_1 \mathbin{\boldsymbol{=}} G_2$. This settles the claim. □

4. Characterizing all subquartic Eulerian graphs of treewidth at most 2

We are now ready to discuss a constructive characterization starting with a simple class of base graphs – the closed necklaces – and then only using the operators $=$ and $\bullet$. More precisely we characterize all Eulerian graphs with treewidth at most 2 and maximum degree 4. A *closed necklace* is a graph which can be constructed from a cycle of length at least 2 by duplicating all of its edges. We define the class $\mathcal{H}$ recursively:

- All closed necklaces are contained in $\mathcal{H}$.
- If $H_1, H_2 \in \mathcal{H}$, then also $H_1 = H_2 \in \mathcal{H}$.

Observation 4.1. *The only biconnected 4-regular graph of treewidth 1 is the closed necklace on two vertices. The only biconnected 4-regular graph on three vertices is the closed necklace on three vertices.*

Lemma 4.2. *Let G be a biconnected 4-regular graph of treewidth 2 which is not isomorphic to a closed necklace. Then G has a 2-cut $\{e_1, e_2\}$ where no end vertex of e_1 coincides with an end vertex of e_2.*

Proof. We prove the following statement by induction on the number of vertices of G: A biconnected 4-regular graph of treewidth 2 is either a closed necklace or it has a 2-cut $\{e_1, e_2\}$ where no end vertex of e_1 coincides with an end vertex of e_2. The base case $|V(G)| \leq 3$ is settled by Observation 4.1. Let now $|V(G)| \geq 4$.

Suppose that G contains a vertex u with $N_G(u) = \{x_1, x_2\}$ such that u is connected to each x_i with exactly two edges. We construct a graph G' by removing u and adding two edges between x_1 and x_2. Observe that G' is still biconnected, 4-regular, of treewidth at most 2. By induction G' is either a closed necklace – in this case G is also a closed necklace. Or G' contains a two-edge-cut of the desired form, then it is also a cut of the desired form in G.

Now suppose that each vertex in G which has exactly two neighbours is connected to one of them with three edges and to the other one with a single edge. Let $(\{X_i : i \in I\}, T)$ be a smooth tree decomposition of G of width 2. Let l be a leaf in T with unique neighbour k, which exists as $\operatorname{tw}(G) = 2$ and $V(G) \geq 4$. As the tree decomposition is smooth we have $X_l = \{u, x_1, x_2\}$ and $X_k = \{v, x_1, x_2\}$ with distinct vertices $u, v, x_1, x_2 \in V(G)$. The biconnectivity of G and the structure of the bags X_l and X_k imply $N_G(u) = \{x_1, x_2\}$. We may assume that there are three edges connecting u to one of its neighbours, say x_1. Let $N_G(x_1) = \{u, x_1'\}$ for some $x_1' \in V(G)$. Note that $x_1' \neq x_2$ as otherwise x_2 would be a cut-vertex, contradicting the fact that G is biconnected. Thus, $\{x_1x_1', ux_2\}$ is a 2-cut of the desired form in G. □

Theorem 4.3. *Let G be a graph. Then $G \in \mathcal{H}$ if and only if it is a biconnected 4-regular graph of treewidth at most 2.*

Proof. Let $G \in \mathcal{H}$. Note that this implies that G is 2-edge-connected by Lemma 3.6. If G is a closed necklace, it is biconnected, 4-regular and fulfils $\operatorname{tw}(G) \leq 2$. We prove that G fulfils the desired properties by induction on the number of vertices. If $V(G) = 2$ it is a closed necklace. So assume that $V(G) \geq 3$ and G is not a closed necklace. Consequently $G = G_1 \boldsymbol{=} G_2$ for two graphs $G_1, G_2 \in \mathcal{H}$. By induction G_1 and G_2 are biconnected, 4-regular and have treewidth at most 2. Then also G is 4-regular, $\mathrm{tw}(G) \leq 2$ by Lemma 3.7 and G is biconnected by Lemma 3.6.

Let now G be biconnected 4-regular of treewidth at most 2. We prove $G \in \mathcal{H}$ by induction on $|V(G)|$. If $|V(G)| \in \{2, 3\}$ then G must be the necklace with two or three vertices by Observation 4.1 and thus $G \in \mathcal{H}$. Let now $|V(G)| \geq 4$. If G is not a closed necklace, then $\mathrm{tw}(G) = 2$ by Observation 4.1. We may apply Lemma 4.2 and obtain that $G = G_1 \boldsymbol{=} G_2$ for suitable G_1, G_2. Observe that G_1 and G_2 are biconnected, cf. Lemma 3.6, and 4-regular. Furthermore, their treewidth is bounded by 2 since they are minors of G. By induction, $G_1, G_2 \in \mathcal{H}$. Thus, also $G \in \mathcal{H}$. □

Let v be a cut-vertex in a 4-regular graph G. The degree of v in the biconnected components of G is 2 since otherwise the edges incident to v would contain an odd cut in an Eulerian graph. Consequently, all degrees of vertices in biconnected components of G lie in $\{2, 4\}$. We conclude that a biconnected component of a 4-regular graph is either a cycle or can be obtained from a biconnected 4-regular graph by subdivision. Together with Theorem 4.3 we obtain:

Corollary 4.4. *A connected graph G is 4-regular of treewidth at most 2 if and only if each of its biconnected components H is either a cycle such that each of its vertices is a cut-vertex in G or the graph obtained by successively resolve all former cut-vertices in H is contained in $\mathcal{H}$.*

We obtain a constructive characterization of the class $\mathcal{H}'$ containing all Eulerian graphs of treewidth at most 2 and with maximum degree 4 in a straightforward way:

- All closed necklaces are in $\mathcal{H}'$.
- All cycles are in $\mathcal{H}'$.
- If $G \in \mathcal{H}'$ and G' is obtained from G by subdividing an edge then $G' \in \mathcal{H}'$.
- If $G_1, G_2 \in \mathcal{H}'$, then $G_1 \boldsymbol{=} G_2 \in \mathcal{H}'$.
- If for $i \in \{1, 2\}$ $G_i \in \mathcal{H}'$ and $v_i \in V(G_i)$ with $\deg_{G_i}(v_i) = 2$, then $(G_1, v_1) \bullet (G_2, v_2) \in \mathcal{H}'$.

Now that we have extensively studied the applications of the binary operator $\boldsymbol{=}$, we continue with considering the class of graphs which arises using the operators $\bullet$ and $\bar{\bullet}$.

5. Constructive characterization of all graphs with unique cycle-decomposition size

In this section we prove our main result – two equivalent characterizations for the class of graphs where the minimum and maximum number of cycles in a cycle decomposition coincide. We show first that the class of graphs with unique cycle decomposition size is contained in the class of graphs where two cycles intersect at most twice.

Lemma 5.1.

(i) Let H be an Eulerian subgraph of an Eulerian graph G. If $c(H) < \nu(H)$ then $c(G) < \nu(G)$.

(ii) Let H' be a graph which is decomposable into two edge disjoint cycles that have more than two vertices in common. Then $c(H') = 2$ and $\nu(H') \geq 3$.

In particular: An Eulerian graph G containing two edge-disjoint cycles that have more than two vertices in common satisfies $c(G) < \nu(G)$.

Proof. Let $\overline{\mathcal{C}}$ be a maximum cycle decomposition of H and $\underline{\mathcal{C}}$ be a minimum cycle decomposition of H. Further let $\mathcal{C}$ be a cycle decomposition of $G - E(H)$. We obtain

$$c(G) \leq |\mathcal{C} \cup \underline{\mathcal{C}}| < |\mathcal{C} \cup \overline{\mathcal{C}}| \leq \nu(G),$$

proving the first claim.

Let now $H' = C_1 \cup C_2$ for two edge-disjoint cycles C_1 and C_2. Further let $v_1, v_2, v_3 \in V(C_1) \cap V(C_2)$ be three distinct vertices. Let $i \in \{1, 2\}$. As C_i is a cycle there exists a path P_i from v_1 to v_2 with $v_3 \notin V(P_i)$, which is a subgraph of C_i. Then P_1P_2 is even and the degree of v_3 in $H' - E(P_1P_2)$ is 4. We get $\nu(H') \geq \nu(H' - E(P_1P_2)) + \nu(P_1P_2) \geq 2 + 1 = 3$ as claimed. □

We are now ready to present a constructive characterization of all Eulerian graphs with the property that the number of cycles in a cycle decomposition is unique. Let us define a class of graphs $\mathcal{G}$, where the base graphs are Eulerian multiedges and all other graphs recursively arise from operations on two disjoint graphs in the class.

- If G is an Eulerian multiedge, i.e. a graph that consist only of two vertices and an even number of parallel edges between the two vertices, then $G \in \mathcal{G}$.
- Let $G_1, G_2 \in \mathcal{G}$ with $V(G_1) \cap V(G_2) = \emptyset$ and v_i a vertex in G_i for $i \in \{1, 2\}$. Then $(G_1, v_1) \bullet (G_2, v_2) \in \mathcal{G}$.
- Let $G_1, G_2 \in \mathcal{G}$ with $V(G_1) \cap V(G_2) = \emptyset$, e_i be an edge in G_i from u_i to v_i for $i \in \{1, 2\}$. Then $(G_1, e_1, u_1) \bar{\bullet} (G_2, e_2, u_2) \in \mathcal{G}$.

Theorem 5.2. *Let G be a graph. The following three statements are equivalent.*

(i) G is Eulerian with $c(G) = \nu(G)$.

(ii) G is Eulerian and no two edge disjoint cycles in G have more than two vertices in common.

(iii) $G \in \mathcal{G}$.

Proof. (i) implies (ii): This implication is stated in Lemma 5.1.

(ii) implies (iii): Suppose that there are graphs satisfying (ii) but not (iii). Then amongst those graphs there exists a graph G of lowest order. Note that G is not an Eulerian multiedge, since these satisfy (iii). We establish further structural properties of G:

Property 1. *G is biconnected.*

Proof of Property 1: Suppose G is not biconnected. Then there exists some cut-vertex $v \in V(G)$. Thus, there are two graphs G_1 and G_2 such that $G = G_1 \bullet G_2$. As no two cycles in G_1 and G_2 have more than two vertices in common, we get $G_1, G_2 \in \mathcal{G}$ by the minimality of G and thereby $G \in \mathcal{G}$, contradicting the choice of G.

Property 2. *For all $e \in E(G)$: $G - e$ is biconnected.*

Proof of Property 2: Suppose that $G - e$ is not biconnected. Then there are two graphs G_1 and G_2 such that $G = G_1 \,\bar{\bullet}\, G_2$. Note that the vertex $v \in V(G)$ that is split up in G_1 and G_2 cannot be an endpoint of e, as G is biconnected by Property 1. Further observe that neither G_1 nor G_2 contain edge disjoint cycles with more than two vertices in common. By the minimality of G we obtain $G_1, G_2 \in \mathcal{G}$. Consequently $G \in \mathcal{G}$. A contradiction.

Property 3. *For all $v \in V(G)$: $G - v$ is 2-edge-connected.*

Proof of Property 3: Suppose that $G - v$ contains a one-edge-separator. Again, there are two graphs G_1 and G_2 such that $G = G_1 \,\bar{\bullet}\, G_2$ and we can argue as in the proof of Property 2.

Property 4. *For all $v \in V(G)$ there is at most one neighbour of v that is connected to v by multiple edges.*

Proof of Property 4: Assume that there is a vertex v that is connected to two different vertices w_1 and w_2 by multiple edges. By Property 3 we know that $G - v$ is 2-edge-connected. By Menger's Theorem (see [12]) there exist two edge disjoint paths P_1, P_2 from w_1 to w_2 in $G - v$. But then the two cycles $vw_1P_1w_2v$ and $vw_1P_2w_2v$ are edge disjoint and share more than two vertices. This is a contradiction to (ii).

Property 5. *For all $v \in V(G)$ we have $|N(v)| \geq 4$.*

Proof of Property 5: Suppose there is a vertex v with $|N(v)| \leq 3$. Assume that $|N(v)| = 1$. Then G is either an Eulerian multiedge or not biconnected – a contradiction to the assumption respectively Property 1. Now assume that $|N(v)| = 2$, say $N(v) = \{w_1, w_2\}$. By Property 4 v cannot be connected to both neighbours by multiple edges, say v is connected to w_1 by a single edge e. If we delete w_2 from $G - e$ we isolate v which is a contradiction to Property 2.

Last assume that $|N(v)| = 3$, say $N(v) = \{w_1, w_2, w_3\}$. By Property 4, we may further assume that v is connected to w_1 and w_2 by a single edge only. By Property 2 the graph $G - vw_1$ is biconnected. Thus, there is a cycle C in $G - vw_1$ containing the vertices v and w_1. Since w_2 and w_3 are the only neighbours of v in $G - vw_1$, we obtain that C also contains the vertices w_2 and w_3. The graph $G - E(C)$ is even and thus vw_1 is contained in some cycle C' in $G - E(C)$. The single edge vw_2 is contained in C. Hence, v has only neighbours w_1 and w_3 in $G - E(C)$. Thus, C' contains the vertex w_3 as well. Thereby C and C' are two edge disjoint cycles with more than two vertices in common – a contradiction.

We now exploit Properties 4 and 5 to complete the proof. Regard a path $P = v_1v_2 \ldots v_k$ with the property that $N(v_k) \subseteq \{v_1, \ldots, v_{k-1}\}$ and $v_1v_k \in E$. Such a path can be found in a greedy fashion: Start at some vertex v in the graph and always move to a new vertex until all neighbours of the current vertex w have already been visited. The resulting path contains the neighbourhood of w. Now simply set v_1 to be the neighbour of w that has been visited first and the subsequent vertices accordingly. By Property 5 we have $|N(v_k)| \geq 4$. Thus, we can find $i, j \in \{2, .., k-2\}$ with $i \neq j$ and $v_i, v_j \in N(v_k)$. Property 4 implies that v_k is connected to v_i or v_j by a single edge. Without loss of generality let this be v_i. Set $C := v_1v_2...v_kv_1$. Then $G - E(C)$ is an even graph and we can find a cycle C' in $G - E(C)$ containing the edge v_kv_i. Since $N(v_k) \subseteq \{v_1, \ldots, v_{k-1}\}$

the two edge disjoint cycles C and C' have more than two vertices in common, which contradicts assumption (ii). Altogether we may conclude $G \in \mathcal{G}$.

(iii) implies (i): Eulerian multiedges fulfil property (i). Let G_i be a graph with $c(G_i) = \nu(G_i)$ for $i \in \{1, 2\}$. If G arises from vertex-identification or vertex-edge-identification from graphs G_1 and G_2, by Lemma 3.1 we have

$$\nu(G) - c(G) = \nu(G_1) - c(G_1) + \nu(G_2) - c(G_2) = 0,$$

which implies (i). □

Combining the constructive characterization in Theorem 5.2 with Lemma 3.7 we obtain that all graphs with unique cycle number are of treewidth at most 2. In particular, they are planar and at most 2-vertex-connected.

6. Algorithmic recognition of graphs with unique cycle number

In this section, we present an $\mathcal{O}(n(m + n))$-algorithm which decides if the cycle number of a given Eulerian graph is unique. The main idea of the algorithm is to exploit the following two observations:

Observation 6.1. *Cycles are subgraphs of the biconnected components of a given graph. Hence: A graph G fulfils $c(G) = \nu(G)$ if and only if this equation holds true for each of its biconnected components.*

Observation 6.2. *Let G be a biconnected graph. Then G fulfils $c(G) = \nu(G)$ if and only if it fulfils one of the following two properties:*

- *The graph G is an Eulerian multiedge.*
- *There exists graphs G_1 and G_2 such that $G = G_1 \bar{\bullet} G_2$. For any two graphs G_1 and G_2 satisfying this equation it holds that $c(G_1) = \nu(G_1)$ and $c(G_2) = \nu(G_2)$.*

Proof. By Theorem 5.2 $G \in \mathcal{G}$. Hence G is either an Eulerian multiedge or there exists $G_1, G_2 \in \mathcal{G}$ with $G = G_1 \bullet G_2$ or $G = G_1 \bar{\bullet} G_2$. The case $G = G_1 \bullet G_2$ cannot occur since G is biconnected. Now assume that G_1, G_2 are graphs with $G = G_1 \bar{\bullet} G_2$. Suppose that $c(G_i) < \nu(G_i)$ for some $i \in \{1, 2\}$. We obtain by Lemma 3.1 that $c(G) = c(G_1) + c(G_2) - 1 < \nu(G_1) + \nu(G_2) - 1 = \nu(G)$. A contradiction. □

These two observations already give an outline of the whole algorithm: We start by computing the biconnected components of the given graph. If a biconnected component is of the form $G_1 \bar{\bullet} G_2$, we replace it by $G_1 \cup G_2$ and check if further decomposition is possible. Corollary 3.5 ensures us that G_1 and G_2 are still biconnected - hence, it suffices to replace $G_1 \bar{\bullet} G_2$ by G_1 and G_2 in the list of biconnected components. If at some point of the algorithm no component allows for further decomposition, the input graph has a unique cycle number if and only if each of the computed components is an Eulerian multiedge.

Definition 6.3 (Vertex-Edge Separation)**.** *Let G be a disjoint union of biconnected graphs. Further let $v \in V(G)$ and $e \in E(G)$ be a vertex and an edge in the same component H of G. We call the tuple (v, e) a* vertex-edge-separator *in G if $H - v - e$ has more than one component. Observe that (v, e) is a vertex-edge-separator if and only if there exist biconnected graphs H_1, H_2 with edges $e_i = u_i v_i \in E(H_i)$ for $i = 1, 2$, such that $H = (H_1, e_1, u_1) \bar{\bullet} (H_2, e_2, u_2)$ where v is the vertex that arises from the identification of v_1 and v_2 and e is the edge from u_1 to u_2 in H. We call the process of replacing H by $H_1 \cup H_2$ in G a* vertex-edge-separation step*. The constructed graph is called* vertex-edge-separation *of G. Observe that the constructed graph is again a disjoint union of biconnected graphs by Corollary 3.5.*

Before we describe the algorithm we will prove a Lemma showing that edges which are not contained in a vertex-edge separator at some step of the algorithm will never be contained in a vertex-edge separator. This proof implies that it suffices to check for each vertex only once whether it is contained in a vertex-edge-separator during the algorithm.

Lemma 6.4. *Let G be a biconnected graph satisfying $G = G_1 \bar{\bullet} G_2$ for two graphs G_1, G_2. Further let $v \in V(G)$ be some vertex in G which is not contained in any vertex-edge separator of G. Then v is not contained in any vertex-edge separator of G_1 or G_2.*

Proof. The graphs G_1 and G_2 both are biconnected and consequently also 2-edge-connected by Corollary 3.5. As v is not contained in a vertex-edge separator in G it is either contained in G_1 or G_2. Hence, by Lemma 3.4 v cannot be contained in a vertex edge separator in G_1 or G_2. □

We are now ready to present a formal algorithm and prove its correctness. In the description of Algorithm 6.1 we use the two *black box procedures* FINDCUTEDGE and SPLIT:

FINDCUTEDGE(G) returns a cut-edge of G if one exists and `Nil` else.

SPLIT(G, v) gets a graph G and a cut-vertex $v \in V(G)$ as input. Let G_1 and G_2 be graphs satisfying $G = (G_1, v_1) \bullet (G_2, v_2)$ where v is the vertex that arises from identifying v_1 with v_2. The procedure returns $G_1 \cup G_2$, v_1 and v_2.

When implementing the algorithm the two procedures would rather be done at the same time using a slightly modified version of the *lowpoint algorithm* for finding biconnected components by Hopcroft and Tarjan, cf. [5]. We merely state it in the presented way to better catch the intuition behind the algorithm.

Theorem 6.5. *Given a biconnected graph G with n vertices and m edges Algorithm 6.2 returns a graph G' that does not contain a vertex-edge separator. The graph G can be obtained from G' by repeated vertex-edge-identification of connected components of G'. Algorithm 6.2 can be implemented to run in time $\mathcal{O}(n \cdot (m + n))$.*

Proof. Note that each time a vertex-edge separation step is applied the number of vertices, edges and components of G all increase by 1. The size of the largest component never increases though and at least one component becomes smaller. Thus, Algorithm 6.2 terminates.

Algorithm 6.1 Test if vertex is contained in a vertex-edge separator, if so apply vertex-edge separation.

TESTANDDECOMPOSE(G, v)

Input: Graph G and vertex $v \in V(G)$.
Output: Graph G, vertices v_1, v_2

$e = u_1u_2 =$ FINDCUTEDGE$(G - v)$
if e is not `Nil` **then**
    $G = G - e$
    $G, v_1, v_2 =$ SPLIT(G, v)
    Add an edge between u_1 and v_1 to G.
    Add an edge between u_2 and v_2 to G.
    return G, v_1, v_2
else
    return G, `Nil`, `Nil`
end if

Algorithm 6.2 Computation of vertex-edge-components using vertex-edge separation.

VE-COMPONENTS(G)

Input: Biconnected graph G.
Output: Disjoint union G of biconnected graphs.

$S := V$
while $S \neq \emptyset$ **do**
    Take out arbitrary $v \in S$
    $G, v_1, v_2 =$ TESTANDDECOMPOSE(G, v)
    if $v_1 \neq$ `Nil` **then**
        Add v_1 and v_2 to S.
    end if
end while
return G

Let G be a biconnected graph and G' the graph returned by the algorithm starting with G. By Lemma 6.4 any vertex that is not contained in a vertex-edge separator in a graph G is also not contained in a vertex-edge separator in $G_1 \cup G_2$ with $G = G_1 \,\bar{\bullet}\, G_2$. As every vertex in G' is at some point contained in the set S and, when regarded, is only kept in the graph if it is not contained in a vertex-edge separator, no vertex of G' can be contained in a vertex-edge separator. This proves the correctness of the Algorithm.

Now we discuss the running time of the algorithm. If G contains only one vertex, the algorithm terminates after the first iteration, as no vertex-edge separation step can be applied. Now assume that $n \geq 2$. Algorithm 6.2 never creates a component with only one vertex. Thereby any component of G' contains at least two vertices. Let k be the number of vertex-edge separation steps taken during the whole procedure. We get that the number of components in G' is exactly $k + 1$, so the number of vertices in G' is at least $2 \cdot (k + 1)$. As in each iteration exactly one additional vertex is added to the graph, we know that $|V(G')| = k+n$. Thus, $k+n \geq 2\cdot(k+1)$ which implies $k \leq n-2$. As already mentioned, we can find a cut-edge and split the graph using a slightly altered version of the usual *lowpoint algorithm* for finding biconnected components, cf. [5]. This algorithm can be implemented to run in time $\mathcal{O}(n + m)$. Thus any call to TESTANDDECOMPOSE needs at most time $\mathcal{O}(n + m)$. Altogether Algorithm 6.2 can be implemented to run in time $\mathcal{O}(n(n + m))$. □

Next we want to use Algorithm 6.3 to find out if the cycle number of a biconnected graph G is unique. As pointed out earlier, we will do this by simply testing if all components remaining, after Algorithm 6.2 has terminated, are Eulerian multiedges.

Algorithm 6.3 Test if cycle number of a graph is unique.

ISCYCLENUMBERUNIQUE(G)

Input: Biconnected graph G.

Output: `True`, if cycle number is unique; `False`, else.

```
H = VE-COMPONENTS(G)
for all v ∈ V(H) do
    if |N(v)| ≠ 1 or # of incident edges is odd then
        return False
    end if
end for
return True
```

Theorem 6.6. *Algorithm 6.3 correctly decides if the cycle number of a biconnected graph G with n vertices and m edges is unique. It can be implemented to run in time $\mathcal{O}(n(m + n))$.*

Proof. First note, that a graph H is a disjoint union of Eulerian multiedges if and only if each vertex has exactly one neighbour and the number of incident edges is even. This proves that Algorithm 6.3 returns `True` if and only if the graph H in the algorithm is a collection of Eulerian multiedges. The running time is clear as the whole algorithm is clearly dominated by the running time of Algorithm 6.2.

It remains to show that this indeed is a necessary and sufficient condition for the graph G to have a unique cycle number. It is easy to see that G is contained in $\mathcal{G}$, as in order to create it we only have to do the algorithm backwards. Now assume that G has a unique cycle number but Algorithm 6.2 does not terminate with a collection of Eulerian multiedges. Then there exists a component H, which is not an Eulerian multiedge and does not allow a vertex-edge separation step. This implies that $H \notin \mathcal{G}$ and by Theorem 5.2 we have $\nu(H) > c(H)$. If we now apply Lemma 3.1 multiple times, we get that $\nu(G) > c(G)$, which is a contradiction to G having unique cycle number. □

Observe that we can also use Algorithm 6.2 to reduce computation of minimum or maximum cycle number to smaller graphs: We simply run the algorithm on a given graph G and compute a minimum respectively maximum cycle decomposition in the outputted components. By Lemma 3.1 we can then puzzle the cycle decomposition together in order to obtain a minimum or maximum cycle decomposition of G.

Acknowledgement

We would like to thank the anonymous reviewer, Sebastian Johann, Sven O. Krumke, Eva Schmidt and Till Heller for their helpful comments on this article.

[1] H.L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth, *Theoret. Comput. Sci.* **209** (1-2) (1998), 1–45.

[2] R. Diestel, *Graph Theory*, Graduate Texts in Mathematics, Springer, 2000.

[3] H. Fleischner, *Eulerian graphs and related topics*, 1990.

[4] I. Holyer. The NP-completeness of some edge-partition problems, *SIAM J. Comput.* **10** (4) (1981), 713–717.

[5] J. Hopcroft and R. Tarjan, Algorithm 447: Efficient algorithms for graph manipulation, *Commun. ACM* **16** (6) (1973), 372–378.

[6] M. Krivelevich, Z. Nutov, M.R. Salavatipour, J.V. Yuster, and R. Yuster, Approximation algorithms and hardness results for cycle packing problems, *ACM Trans. Algorithms* **3** (4) (2007).

[7] S.O. Krumke and H. Noltemeier, *Graphentheoretische Konzepte und Algorithmen* (2009).

[8] W. Mader. Konstruktion aller n-fach kantenzusammenhañgenden digraphen, *European J. Combin.* **3** (1) (1982), 63–67.

[9] C. Otto and P. Recht, Maximum cycle packing using SPR-trees, *Electron. J. Graph Theory Appl.* **7** (1) (2019), 147–155.

[10] B. Péroche. NP-completeness of some problems of partitioning and covering in graphs, *Discrete Appl. Math.* **8** (2) (1984), 195–208.

[11] W.T. Tutte, A theory of 3-connected graphs, *Indag. Math.* **23** (1961), 441–455.

[12] D.B. West, *Introduction to Graph Theory*, 2001.