AKCE J. Graphs. Combin., 4, No. 1 (2007), pp. 61-82

On Simply Structured Kernel Bases of Unicyclic Graphs

T. Sander and J. W. Sander
Institut für Mathematik
Technische Universität Clausthal
D-38678 Clausthal-Zellerfeld, Germany.
e-mail: { *torsten|juergen* } *.sander@math.tu - clausthal.de*

Communicated by: Matthias Kriesell
Received 15 March 2006; accepted 10 January 2007

Abstract

In this paper all those unicyclic graphs are characterized for which there exists a basis of the kernel that consists only of vectors with entries from $\{-1, 0, 1\}$. Three different characterizations are obtained, based on an algorithmic, an algebraic, and a structural criteria, respectively. Algorithmic construction of such bases is discussed as well.

Keywords: unicyclic, kernel, null space, basis, Gaussian elimination.

2000 Mathematics Subject Classification: Primary 05C50, Secondary 15A18.

1. Introduction

Considering real vector spaces it is sometimes desirable to find a basis of a particular structure, e.g. an orthogonal basis or a basis of vectors whose components are integers. In particular, let us say that a basis is simply structured if it consists only of vectors with entries in $\{-1, 0, 1\}$.

Beside the fact that it is a structural feature of a vector space to have a simply structured basis, such a basis can have significant computational advantages. Apart from a certain degree of sparsity caused by the zero entries it may also be possible to use the nonzero entries to advantage: The vectors of a simply structured basis of a tree kernel can be compressed into bit fields by simply discarding the entry signs, because it is possible to reconstruct the distribution of signs [10].

Simply structured bases of special vector spaces, namely kernels or other eigenspaces of matrices describing finite graphs have attracted some attention in recent years. For the kernel of the incidence matrix of a graph it has been shown that it always admits a basis with entries from the set $\{-2, -1, 0, 1, 2\}$ (cf. [14], [2]). For a graph without cut-vertices [8] or only without cut-edges [2] this set can be reduced to $\{-1, 0, 1\}$, yielding a simply

structured kernel basis. It is even possible to characterize all graphs whose incidence matrix admits a simply structured kernel basis [2].

Let us turn to the kernel of the adjacency matrix of a graph (and hereafter refer to it as the kernel of the graph). Here, the situation is much harder. Current research has not progressed as far as to yield a complete characterization of graphs with simply structured bases. There exists, however, a number of results for certain graph classes. For example, the kernel (and the eigenspace of eigenvalue -1) of a graph without induced path P_4 (a so-called cograph) always contains a simply structured basis [11]. Further, in [10] and, independently, in [1] it has been proven that every forest has a simply structured kernel basis. Other results about graphs with eigenvalue zero include e.g. [4], [12], [5], [13].

This paper is dedicated to the study of unicyclic graphs. It is easy to find unicyclic counterexamples that do not admit a simply structured kernel basis. Weakening the demands for a common kernel basis property, it has been proven in [1] that for every unicyclic graph there exists a labelling of its vertices such that the respective row-reduced echelon form is integral. In contrast to this, we are interested in identifying exactly those unicyclic graphs that do admit a simply structured kernel basis. We will utilize the results and methods presented in [10].

Our main result takes the following form. Let G be a unicyclic graph with cycle C and let T_i be the trees emanating from C. Assume that each tree T_i is attached to C by an edge v_iu_i such that u_i and v_i are nodes of C and T_i, respectively. Then G has a simply structured kernel basis unless $|C| \equiv 2 \mod 4$ and a particular condition holds. We derive three different conditions, allowing to state an algorithmic, an algebraic, and a structural characterization of the unicyclic graphs with simply structured kernel bases.

2. Graph eigenvectors

Let us recall some basics of algebraic graph theory, in particular the notion of graph eigenvectors and eigenvalues (cf. [7]). The eigenvalues of a graph G with r nodes are the eigenvalues of its adjacency matrix $A(G) \in \mathbb{R}^{r\times r}$. Note that this definition is invariant under isomorphism so that the eigenvalues of a graph do not depend on the numbering of its nodes.

An eigenvector x of G for eigenvalue λ is given by a nonzero solution of the equation $A(G)x = \lambda x$. This definition depends on the node numbering of the graph. Observe, however, that every vector $x = (x_i) \in \mathbb{R}^r$ can be represented by assigning weights to the nodes of G such that x_i is the weight of node v_i, given that $V(G) = \{v_1, \dots, v_r\}$. This leads to a graph eigenvector definition that does not depend on the node order. Therefore, each time we determine an eigenvector of an adjacency matrix, we implicitly consider it as a function $V(G) \to \mathbb{R}$.

Reading the equation $A(G)x = \lambda x$ for each entry of x separately, we see that we may check if a given vector x is an eigenvector of G for eigenvalue λ as follows. Introduce node weights on G according to vector x and verify that for every node of G the sum of

the weights of its neighbors equals λ times its own weight. This procedure will be called the summation rule.

3. The FOX algorithm

To construct the kernel basis of a graph we usually perform Gaussian elimination on the adjacency matrix of the given graph. Since we aim to construct a basis that exhibits certain properties there will typically arise additional rules to observe.

In this section we present an elimination algorithm that performs a possibly incomplete Gaussian elimination scheme. Its pivoting strategy only considers unit vector rows of the coefficient matrix. Note that for an adjacency matrix this only turns some nonzero entries into zero entries.

For a formal analysis we want to avoid excessive use of matrix indices and therefore seek a presentation of the algorithm that carries out the elimination operations directly on a directed graph. If an undirected graph is given, then this approach uses the directed graph with the same adjacency matrix as the given undirected graph. Therefore it is clear that the described elimination procedure only removes edges from the digraph.

Let $G = (V, E)$ be a connected graph. Construct the corresponding bidirectional orientation $\hat{G}$, i.e. the unique digraph that has the same adjacency matrix as G.

We will introduce an algorithm that takes the graph $\hat{G}$ as input and produces a labeled subgraph $\tilde{D}$ of the input graph. Each node will be tagged with at least one of the three labels `F`, `O` and `X`.

Algorithm 3.1. (FOX Algorithm)

(1) Let all nodes of $\hat{G}$ be untagged.

(2) Find a node v without `X` *tag that has exactly one outgoing edge e. If no such node exists, go to step (6).*

(3) Tag node v with `X`.

(4) Let w be the unique node with $e = vw$. Except for e, remove all incoming edges of w.

(5) Tag node w with `O` *and go to step (2).*

(6) Tag all untagged nodes with `F`.

Since for each run of the main loop from step (2) to (5) another node becomes `X` tagged, it is clear that the algorithm stops after at most $|V|$ iterations of the main loop.

Let $H_{\{\text{labels}\}}$ be the subgraph of $\tilde{D}$ induced by all nodes that have each been tagged with all of the labels mentioned in the subscript.

We will say that v is an $\mathtt{O}$ tagged node if among its tags there is an $\mathtt{O}$ label. Conversely, we require an $\overline{\mathtt{O}}$ tagged node to have no $\mathtt{O}$ labeled tag.

It is obvious that every node of $\tilde{D}$ has been tagged.

Algorithm 3.1 was first presented in [10], where it was used to study the kernel of trees. The remainder of this chapter consists of results on Algorithm 3.1 that we cite from [10] and two new or extended results (Lemmas 3.13 and 3.18).

Let us summarize some elementary properties of the output graph $\tilde{D}$:

Lemma 3.2. $\ker G = \ker \tilde{D}$.

Lemma 3.3.

1. *Every* $\mathtt{X}$ *tagged node of* $\tilde{D}$ *has exactly one outgoing neighbor, namely an* $\mathtt{O}$ *tagged node.*
2. *Every* $\mathtt{O}$ *tagged node of* $\tilde{D}$ *has exactly one incoming neighbor, namely an* $\mathtt{X}$ *tagged node.*
3. *Every* $\mathtt{X}$ *tagged node has at least one incoming neighbor.*
4. *Every weak component of* $H_{\mathtt{XO}}$ *is a 2-cycle and also a weak component of* $\tilde{D}$.
5. *If* v_1 *is an* $\mathtt{F}$ *tagged node of* $\tilde{D}$ *and* v_1v_2 *is an outgoing edge of* v_1, *then* v_2 *is an* $\mathtt{F}$ *tagged node as well and* v_2v_1 *is also an edge of* $\tilde{D}$.

Lemma 3.4. *Let* v *be an* $\mathtt{F}$ *tagged node and let* H *be the strong component of* $\tilde{D}$ *that contains* v. *Then all adjacencies between nodes of* H *are mutual.* H *contains only* $\mathtt{F}$ *tagged nodes. Further,* $|H| = 1$ *or* $|H| \geq 3$. *In the second case* H *contains a bidirectional cycle with at least* 3 *nodes.*

Incoming neighbors of H *can only be* $\overline{\mathtt{XO}}$ *tagged nodes.* H *has no outgoing neighbors.*

Corollary 3.5. *Let* G *be a tree. Then* $H_{\mathtt{F}}$ *contains only isolated points.*

Lemma 3.6. *Assume that* $H_{\mathtt{F}}$ *contains only isolated points.*

Then the $\mathtt{F}$ *tagged nodes of* $\tilde{D}$ *are exactly the* $\overline{\mathtt{X}}$ *tagged nodes that have no outgoing edges. Incoming edges of* $\mathtt{F}$ *tagged nodes always start from* $\overline{\mathtt{XO}}$ *tagged nodes.*

Lemma 3.7. *If the FOX algorithm is conducted in a way that avoids the assignment of multiple labels for as long as possible, then the set of unlabeled nodes at the time of the first assignment of a second tag to an already labeled node is identical to the set of nodes that become* $\mathtt{F}$ *tagged at the end of the FOX algorithm.*

In the following, we will assume that given a graph G we will automatically apply the FOX Algorithm 3.1 to its bidirectional orientation $\hat{G}$ to get the final digraph $\tilde{D}$. The number of F tagged nodes of $\tilde{D}$ will be denoted by $k = |H_{\mathtt{F}}|$.

Lemma 3.8. *Let G be a graph. Then the nodes of $H_{\mathtt{O}}$ obtained after a run of the FOX algorithm on G form a subset of the nodes whose value is zero for every vector from the kernel of G.*

For a forest we can strengthen Lemma 3.8:

Lemma 3.9. *Let G be a forest. Then the graph $H_{\mathtt{O}}$ contains exactly those nodes on which every vector from the graph kernel vanishes. In particular, this node set is invariant for all possible results of the FOX algorithm.*

For any X $\overline{\mathtt{O}}$ tagged node v that has only one incoming edge let $\tilde{R}_v$ be the subgraph of $\tilde{D}$ spanned by the nodes of all directed paths in $\tilde{D}$ starting from v. Likewise, for any F tagged node w let $\tilde{S}_w$ be the subgraph of $\tilde{D}$ spanned by the nodes of all directed paths in $\tilde{D}$ ending at w. The undirected counterparts of $\tilde{R}_v$ and $\tilde{S}_w$ in G are called R_v and S_w, respectively. We call a directed path maximal if it is not contained in any other directed path. Note that the previous definitions apply to arbitrary graphs, not only to trees or forests.

The following result investigates the structure of the subgraphs we have just defined:

Lemma 3.10. *Let G be a graph. Then:*

1. *S_w contains exactly one F tagged node.*
2. *Each maximal directed path in $\tilde{R}_v$ that begins at v starts with an even number of consecutive nodes tagged X $\overline{\mathtt{O}}$ and $\overline{\mathtt{X}}$O alternatively and terminates with an F tagged node.*
3. *Each maximal directed path in $\tilde{S}_w$ that leads to w starts with an even number of consecutive nodes tagged X $\overline{\mathtt{O}}$ and $\overline{\mathtt{X}}$O alternatively and terminates with an F tagged node.*
4. *If G is a tree, then the graphs R_v and S_w each induce a subtree of G.*

Lemma 3.11. *Let G be a tree. Then for every X $\overline{\mathtt{O}}$ tagged node v of $\tilde{D}$ there exists a node w of $\tilde{D}$ such that v lies in $\tilde{S}_w$.*

Lemma 3.12. *Let G be a tree and x a vertex of G. Then, all type $\overline{\mathtt{X}}$O nodes of S_x have degree 2 within this subgraph of G. Consequently, all nodes of S_x whose degree is at least 3 are necessarily of type X $\overline{\mathtt{O}}$ or F.*

The next lemma states that within a subgraph S_w any $\overline{\texttt{O}}$ tagged node may play the part of the F tagged node.

Lemma 3.13. *Let G be a tree and w a node that gets* F *tagged by a given FOX run $\mathcal{A}$. Further, let v be an* X $\overline{\texttt{O}}$ *tagged node of S_w.*

Then there exists a FOX run $\mathcal{B}$ on G that creates the same node tags as run $\mathcal{A}$, except that node v gets F *tagged and w becomes* X $\overline{\texttt{O}}$ *tagged. Moreover, the subgraphs $S_w^{(\mathcal{A})}$ and $S_v^{(\mathcal{B})}$ of G are identical.*

Proof. We may assume w.l.o.g. that $\operatorname{dist}_G(v, w) = 2$, otherwise iterate the proof technique.

According to Lemma 3.10 the node u that lies between v and w is O tagged. Clearly, u must be the partner of v. Let $u_1, \ldots, u_r$ be the other incoming neighbors of w in $\tilde{D}$, all being type O.

Since w is F tagged we see that the operations carried out by FOX run $\mathcal{A}$ may be rearranged such that first only the branches that are connected to w via $u_1, \ldots, u_r$ get tagged and then only the branch connected via u. This does not change the result $\tilde{D}$ or any node tags. Now, when the FOX algorithm would select v we pick w instead. This is possible because the neighbors $u_1, \ldots, u_r$ have all been O tagged so that its only outgoing neighbor is u. Since v was eligible for X tagging one step before and the O tagging of u has removed the edge vu we see that v now becomes F tagged. Denote this altered FOX run by $\mathcal{B}$.

Obviously, except for the edges between the nodes u, v, w and their tags there is no difference between the results of runs $\mathcal{A}$ and $\mathcal{B}$. Since there exits a directed path from w to v in $\tilde{D}^{(\mathcal{B})}$ and u has no other incoming edges other than from w it is clear that $S_w^{(\mathcal{A})}$ and $S_v^{(\mathcal{B})}$ are the identical subgraphs of G. □

Using the following construction it is possible to obtain a simply structured tree kernel basis:

Construction 3.14. *Let G be a tree with r nodes. Assign weight 0 to all* O *tagged nodes of $\tilde{D}$ and also to all* F *tagged nodes except one node w. Assign weight 1 to w and construct $\tilde{S}_w$.*

All X $\overline{\texttt{O}}$ *tagged nodes that are not contained in $\tilde{S}_w$ receive weights of 0. Conducting an incoming edge breadth first search on $\tilde{S}_w$ with starting node w we can assign weights to the remaining* X *tagged nodes as follows. Let x be an unweighted* X *tagged node to be processed and let y be its outgoing* O *tagged neighbor. Then assign to x the negative sum of the weights of all non-incoming neighbors of y.*

Performing this procedure for every F *tagged node of $\tilde{D}$ we obtain a set of k vectors $z_i \in \mathbb{R}^r$.*

Theorem 3.15. *Let G be a tree. Apply Algorithm 3.1 on its bidirectional orientation $\hat{G}$ to get the final digraph $\tilde{D}$.*

Then $k = |H_{\mathtt{F}}|$ equals the degree of singularity of G.

For $k \geq 1$ Construction 3.14 yields a simply structured basis of $\ker G$.

For $k = 0$ the graph $\tilde{D}$ consists only of `XO` *tagged nodes. The perfect matching of G can be constructed by pairing the nodes according to the 2-cycles in $H_{\mathtt{XO}}$.*

With respect to the case $k = 0$ it is useful to note that a tree has a perfect matching if and only if it is nonsingular since the rank of the adjacency matrix of a tree is twice the size of a maximum matching [4], [9]. A perfect matching of a tree is unique.

To conclude this section we present an extension lemma that will be used later on:

Lemma 3.16. *Let G be a tree and v be an* `O` *tagged node of G. Construct a tree G' by connecting v to a new node w.*

Setting the weight of w to zero it is then possible to embed every vector from the kernel of G such that a vector from the kernel of G' is obtained.

Moreover,

$$\dim \ker G' = \dim \ker G + 1$$

and there exists a basis of $\ker G'$ such that all but one vector have zero weights on w.

Remark 3.17. *Note that the previous result can be readily extended to forests.*

In the following, let D be the undirected counterpart of $\tilde{D}$. Then we may extend Lemma 3.2 as follows:

Lemma 3.18. *It holds $\ker G = \ker \tilde{D} = \ker D$ for any graph G.*

Proof. The equality $\ker G = \ker \tilde{D}$ directly follows from the construction of $\tilde{D}$. Now consider the FOX run on G whose result is $\tilde{D}$. Let $e = xy$ be an edge that lies in G but not in the subgraph D. Then according to the definition of the FOX algorithm e can only join `O` tagged nodes. Further, according to Lemma 3.3 the `X` partner of x cannot be y and vice versa. It follows that the considered FOX run on G could be performed in the same way even if the edge e did not exist. Consequently, there exists a FOX run on D that yields $\tilde{D}$. □

4. FOX on Unicyclic Graphs

For graphs that contain cycles it easy to find counterexamples — even from the class of unicyclic graphs — that do not admit a simply structured kernel basis, cf. Figure 1. It shows an example graph and a basis of its one dimensional kernel.

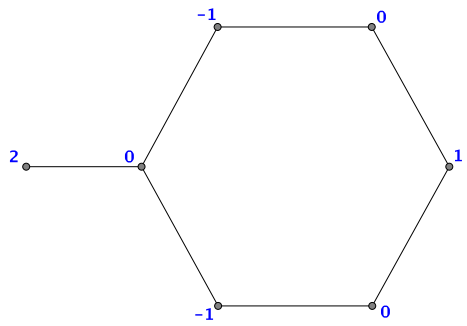

Figure 1: Graph without a simply structured kernel basis

Consider an induced cycle C of a given graph G. Perform the FOX algorithm on G to get $\tilde{D}$ and transfer the node labels to the corresponding nodes of G. We say that C has been *cracked* by this run of FOX if not all of its nodes have been F tagged.

Observation 4.19. *Let G be a unicyclic graph with cycle C and suppose that C gets cracked by a particular FOX run $\mathcal{A}$. Then, clearly, the first node v of C to become $\overline{\mathtt{F}}$ tagged by run $\mathcal{A}$ must get O tagged from outside C, i.e. there exists an X tagged node w in a tree T emanating from C such that v is the outgoing neighbor of w.*

Lemma 4.20. *Let G be a unicyclic graph with cycle C. A given FOX run on G does not crack C if and only if all neighbors of C in G have been O tagged.*

Proof. Assume that a node v from C has a neighbor u in $G - C$ that is either $\mathtt{X}\,\overline{\mathtt{O}}$ or F tagged. Since neither of them is O tagged, u and v must be mutually adjacent in $\tilde{D}$.

Now assume that u is $\mathtt{X}\,\overline{\mathtt{O}}$ tagged. According to Lemma 3.3 the node v must be its only outgoing neighbor so that by Algorithm 3.1 the node v is O tagged, a contradiction.

Let therefore u be F tagged. Let T be the tree in G that is attached to C by the edge uv. Not all neighbors of u within T can be $\overline{\mathtt{XO}}$ tagged since then u would have exactly one outgoing neighbor and would have become X tagged by FOX. So we may assume that u has a neighbor in T that is either $\mathtt{X}\,\overline{\mathtt{O}}$ or F tagged. As above we see that the case $\mathtt{X}\,\overline{\mathtt{O}}$ is impossible.

Continuing this argument, we may conclude that there exists a chain of F tagged nodes starting from v and leading to a leaf of T. But by Lemma 3.3 in $\tilde{D}$ this leaf is mutually adjacent to its neighbor and therefore has only one outgoing neighbor in $\tilde{D}$. Hence, it would have been X tagged by FOX, a contradiction.

The converse statement follows from Observation 4.19. □

Lemma 4.21. *Let G be a unicyclic graph with cycle C. Suppose that C has been cracked by a particular run of FOX. Then C gets cracked by every possible run of FOX.*

Proof. Suppose that there exists a FOX run $\mathcal{A}$ that cracks C and that C has not been cracked by another FOX run $\mathcal{B}$. Determine nodes v, w and a tree T according to Observation 4.19. Rearrange both runs $\mathcal{A}$ and $\mathcal{B}$ in such a way that they start on the nodes of the tree $T + v$. This does not change the resulting digraphs whose node labels we need to consider.

Now focus only on the graph $T + v$. The starting operations of run $\mathcal{A}$ represent a partial FOX run on $T + v$ which may be completed in a way such that v, w becomes an `XO` tagged pair. On the other hand, the operations of run $\mathcal{B}$ represent a complete FOX run on $T + v$ that creates an `F` tag for v. Hence, we have a contradiction to Lemma 3.9. □

As a consequence of the previous lemma, we say that the cycle C of a unicyclic graph G is either cracked or uncracked.

The following investigations will treat these cases separately. To begin with, we consider uncracked unicyclic graphs.

Lemma 4.22. *Let G be a unicyclic graph with uncracked cycle C. If $x \in \ker G$, then $x_C \in \ker C$ holds for the restriction x_C of x to C.*

Proof. According to Lemma 4.20 the neighbors of C in $G - C$ are all `O` tagged. Therefore, the respective node weights must be zero for every vector from the kernel of G. Consequently, we see that on the nodes of C the eigenvector summation rules for C and G coincide. □

Theorem 4.23. *Let G be a unicyclic graph with uncracked cycle C. If the size of C is not a multiple of four, then G has a simply structured kernel basis.*

Proof. Since the size n of C is not divisible by four, the cycle graph C_n is nonsingular [3]. Therefore, according to Lemma 4.22 every vector from $\ker G$ must vanish on the vertices of C. Consider the forest $G - C$ and let T be one of its trees. Then we see by Lemma 4.20 that $x_T \in \ker T$ must hold for the restriction x_T of any vector $x \in \ker G$ to T. Hence, a simply structured kernel basis of G is obtained by means of trivial embedding after using Construction 3.14 to determine simply structured kernel bases for the trees of $G - C$. □

Lemma 4.24. *Let n be a multiple of four and let $x \in \ker C_n$. Then x takes values $c, 0, -c, 0, c, 0, -c, \ldots$ for some $c \in \mathbb{R}$ on consecutive vertices of C_n.*

Proof. This is a well-known result that follows directly from the summation rule. □

Theorem 4.25. *Let G be a unicyclic graph with uncracked cycle C. If the size of C is a multiple of four, then G has a simply structured kernel basis. Moreover,*

$$\dim\ker G = \dim\ker(G - C) + 2.$$

Proof. Since the size n of C is divisible by four, the cycle graph C_n is singular. Therefore, according to Lemma 4.22 every vector from $\ker G$ takes values $c, 0, -c, 0, c, 0, -c, \ldots$ for some $c \in \mathbb{R}$ on the consecutive vertices of C. If $c = 0$, then we may proceed as in the proof of Theorem 4.23. For $c \neq 0$ we deduct from Lemma 4.24 that there are only two linearly independent choices of possible weights on C (the zero-nonzero pattern is rotated by one position). We may assume that these weights are only from the set $\{0, 1, -1\}$. Given such weights on C we need to construct a valid vector from $\ker G$. Assign zero weights to the vertices of all trees emanating from the zero weight vertices of C. Because of Lemma 4.20 we may employ Lemma 3.16 to construct a suitable eigenvector for each tree emanating from a vertex with weight ± 1.

This construction yields $\dim\ker(G - C) + 2$ linearly independent vectors from $\ker G$. These vector even form a basis of $\ker G$. To see this, it suffices to note that B cannot contain two vectors that do not vanish on C but have the same zero-nonzero pattern on C because otherwise a suitable linear combination would yield a vector extended from $\ker(G - C)$. □

An example that illustrates Theorem 4.25 can be found in Figure 2.

Next we consider the case with cracked cycle. It turns out that it demands more effort than the previous case.

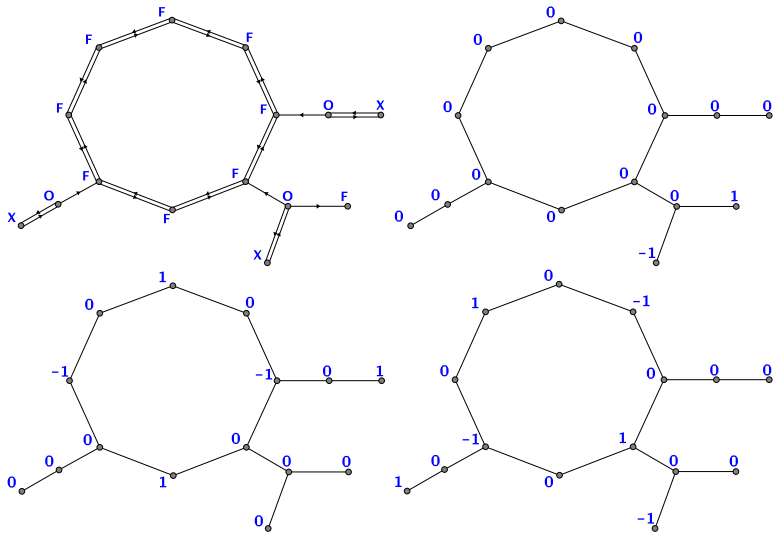

Figure 2: Kernel basis of a unicyclic graph according to Theorem 4.25

Our next theorem determines the size of the kernel of a unicyclic graph with cracked cycle in terms of the number of F tagged nodes. Thus, the result relates to Theorem 3.15.

But beforehand, we require the definition of a trivial embedding function ι. Let H be a subgraph of G and let $w : H \to \mathbb{R}$ be a node weight function on H. Then, $\iota(w) : G \to \mathbb{R}$ is defined by

$$\iota(w)(x) = \begin{cases} w(x) & \text{if } x \in V(H) \\ 0 & \text{else} \end{cases}.$$

Naturally, we assume that ι works accordingly for weight vectors (with respect to a fixed vertex numbering).

Theorem 4.26. *Let G be a unicyclic graph with cracked cycle C. Let therefore u be an O tagged node of C whose X tagged partner v does not lie on C. Then,*

$$\ker G \subseteq \iota(\ker(G - u))$$

and

$$|H_{\mathtt{F}}| = \dim \ker G = \dim \ker(G - u) - 1.$$

Proof. There exists a FOX run on G such that u is the first node of C that receives a tag. Consider the situation just after u has been O tagged. Since the selection of a node to be X tagged does not depend on its incoming edges it is clear that the outgoing edges of u leading to still untagged nodes have no effect on the remaining steps of the FOX run considered. We may therefore remove the corresponding edges from G and still get the same node tags. If we further disconnect u and v we see that v has no more incoming neighbors and must therefore become F tagged. Consequently, there exists a FOX run on $G - u$ that creates the same node tags as on G before, only that v is F tagged. Since $G - u$ is a forest it follows from Theorem 3.15 that the number of F tagged nodes of G exceeds the dimension of $\ker(G - u)$ by exactly one.

Since u is O tagged it holds necessarily that $x_{G-u} \in \ker(G - u)$ for every vector $x \in \ker G$. It follows with Lemma 3.8 that $\ker G$ consists of exactly those vectors $x \in \iota(\ker(G - u))$ that obey the summation rule for the node u. The summation rule poses an at most one-dimensional restriction on $\iota(\ker(G - u))$. If the summation rule did not pose a restriction, then every vector from $\ker G$ would have to be zero on the neighbors of u, in particular on v. But according to Theorem 3.15 there exists a vector from $\iota(\ker(G - u))$ that does not vanish on v, a contradiction. □

Note that Theorem 4.26 does not yet guarantee the existence of simple kernel bases since the component eigenvectors may have to be multiplied to suit the summation rule at the node u.

At this point we can provide a generalization of Lemma 3.9:

Lemma 4.27. *Let G be a unicyclic graph. Then the set $V(H_{\mathtt{0}})$ is invariant for all possible runs of FOX on G and contains exactly those vertices on which every vector from* $\ker G$ *vanishes.*

Proof. From the proofs of Theorems 4.23, 4.25 and 4.26 it follows that for every $\overline{\mathtt{0}}$ tagged node (with respect to a given FOX run) there exists a vector from the kernel of G that does not vanish on this node. Recalling Lemma 3.8, the result follows. □

Continuing our analysis of the cracked case we will now investigate node separation issues since these lead to further subcases we need to consider.

Lemma 4.28. *Let G be a unicyclic graph with cracked cycle C. Then the following statements are equivalent for a given FOX run:*

1. *No two adjacent nodes of C are assigned* $\mathtt{0}$ *tags.*
2. *Adjacent nodes of C do not become separated in $\tilde{D}$.*

Proof. Nodes on C that become separated in $\tilde{D}$ must necessarily be $\mathtt{0}$ tagged since their incoming edges get deleted.

Conversely, assume that there exists a pair x, y of adjacent $\mathtt{0}$ tagged nodes on C. We will show that there exist two nonadjacent nodes in $\tilde{D}$ whose counterparts in C are adjacent.

Case 1. Suppose that both x and y are $\mathtt{X}$ tagged as well. Let y' be the second neighbor of x on the cycle C. Then according to Lemma 3.3 the node x and either y or y' belong to different weak components of $\tilde{D}$.

Case 2. Suppose that at most one of the nodes x and y is $\mathtt{X}$ tagged. Then by Lemma 3.3 the nodes x and y belong to different weak components of $\tilde{D}$. □

The previous lemma motivates the following definition. We say that a cycle C of a graph G is cut by a given FOX run if there exist two nodes of C that are adjacent in G but disconnected in $\tilde{D}$.

It follows directly from Lemma 4.27 that cutting the cycle does not depend on the particular FOX run so that it makes sense to say that a unicyclic graph either has a cut or uncut cycle. A cut cycle is necessarily cracked.

Note that if an even cycle gets cut, then separation occurs for at least two pairs of neighbors since between the separated nodes there must be alternating $\mathtt{0}$ and $\overline{\mathtt{0}}$ tags along the nodes of the cycle.

Corollary 4.29. *Let G be a unicyclic graph with cut cycle C. Then G has a simply structured kernel basis.*

Proof. By Lemma 4.28 we see that D is a forest so that the result follows from Lemma 3.18 and Theorem 3.15. □

Corollary 4.30. *Let G be a unicyclic graph with uncut cracked cycle C. Then C is even.*

Proof. According to Lemma 4.28 the nodes of C do not become separated in $\tilde{D}$. Thus, O tagged and $\overline{\texttt{O}}$ tagged nodes must alternate on C so that C must necessarily be even. □

Corollary 4.31. *Let G be a unicyclic graph with cracked odd cycle C. Then G has a simply structured kernel basis.*

Let us take a closer look at unicyclic graphs with uncut cracked cycles. Before we analyze their kernel structure let us determine when a cracked cycle remains uncut.

We say that the cycle C of a unicyclic graph G can be cracked at node u by node v if u lies on C, v lies outside C and there exists a FOX run on G such that v is the X tagged partner v of u.

Lemma 4.32. *Let G be a unicyclic graph with cracked cycle C. Then C remains uncut if and only if C is even and each pair of nodes at which C may be cracked has even distance in G.*

Proof. Sufficiency follows directly from the proof of Corollary 4.30 and Lemma 4.27.

Conversely, construct a FOX run on G that avoids the creation of X tagged nodes on C for as long as possible. It is obvious that when the first node of C becomes X tagged there are no more untagged nodes on the tree emanating from C so that C has been cracked at all possible nodes. Note that between any two O tagged nodes of C there lies an even number of untagged nodes.

Subsequently, let the FOX run avoid tagging nodes outside C for as long as possible by repeating the following procedure. On C select an arbitrary untagged neighbor v of an O tagged node. If v has no outgoing neighbor, pick again. Otherwise v has exactly one outgoing neighbor, which by construction must be its second neighbor y on C. To see this it suffices to note that since C cannot be cracked at v its neighbors outside C must be O tagged. Consequently, let FOX select v for X tagging. Thus we have reduced the gap of untagged nodes between two particular O tagged nodes on C by two.

It is clear that this procedure leads to a strict O-$\overline{\texttt{O}}$ pattern on C that does not cut C. Employing an argument similar to that in Lemma 3.7 it is guaranteed that the final steps of the FOX run may introduce multiple tags but does not further change the set of O tagged nodes. □

Theorem 4.33. *Let G be a unicyclic graph with uncut cracked cycle C. If the size of C is a multiple of four, then G has a simply structured kernel basis. Moreover, this basis may be chosen to contain at most one vector x with $x_C \in \ker C \setminus \{0\}$.*

Proof. Let u be an `O` tagged node of C whose `X` tagged partner v does not lie on C. Since $G-u$ is a forest, we may use Construction 3.14 to obtain a simply structured kernel basis B of $G-u$. We assume that the FOX run on $G-u$ needed for the construction has been derived from a FOX run on G as described in the proof of Theorem 4.26, yielding the same node labels except for v. Let $B = \{b_0, \ldots, b_k\}$ with $k = |H_{\mathtt{F}}|$, cf. Theorem 4.26. By construction, the restriction of B to $H_{\mathtt{F}}$ yields the standard unit basis. Assume w.l.o.g. that b_0 has value 1 on v. Since G is unicyclic it follows that one component of $G-u$ contains exactly two neighbors of u in G whereas all other components contain at most one neighbor. Therefore, every vector b_i may have at most two nonzero weights on the neighbors of u in G.

We will now construct a simply structured linearly independent subset $B' = \{b'_1, \ldots, b'_k\}$ of $\ker G$. Theorem 4.26 then ensures that B' is a basis of $\ker G$. Consider each vector $b_1, \ldots, b_k$ separately and let n_i be the number of nonzero weights among the neighbors of u in G for the vector b_i.

Case $n_i = 1$**.** Let y_1 be the only neighbor of u with nonzero weight (assume w.l.o.g. that this weight equals one). Then $\iota(b_i - b_0) \in \ker G$. Let therefore $b'_i = \iota(b_i - b_0)$.

Case $n_i = 2$**.** Let y_1, y_2 be the neighbors of u with nonzero weight. Because of Construction 3.14 both y_1 and y_2 lie in the same subgraph S_w of $G-u$ for some `F` tagged node w.

But since the `X` tagged partner v of u does not lie in C it follows that there exists no directed path from y_1 to y_2 via u in $\tilde{D}$ and vice versa. So $C-u$ is necessarily a subgraph of S_w because otherwise the propagation of weights starting from w could not have succeeded. Since the size of C is a multiple of four, we deduce from Construction 3.14 that $\mathrm{dist}_{G-u}(y_1, y_2) \equiv 2 \mod 4$ so that b_i assigns opposite weights to y_1 and y_2. This implies $\iota(b_i) \in \ker G$. Let therefore $b'_i = \iota(b_i)$.

Now suppose that B' contains two nonzero vectors b_{i_1} and b_{i_2} whose restriction to C lies in $\ker C$. It follows from Lemma 4.24 and Lemma 3.8 that there exists a linear combination b' of b_{i_1} and b_{i_2} that vanishes on C. Substitute b' for b_{i_2} in B'. Clearly, B' retains its basis property. Repeat the procedure until B' only contains at most one vector x with $x_C \in \ker C \setminus \{0\}$. □

We have seen in the proof of Theorem 4.33 that there may exist an `F` tagged node w such that for some `O` tagged node u from C its neighbors y_1 and y_2 in C both lie in S_w. We may ask what happens if there exist several such nodes w, cf. Figure 3.

Lemma 4.34. *Let G be a unicyclic graph with cracked cycle C. Let u be an `O` tagged node of C whose `X` tagged partner v does not lie on C and let y_1 and y_2 be the neighbors*

of u in C.

If there exist distinct nodes w_1 and w_2 such that y_1 and y_2 both lie in S_{w_1} and S_{w_2}, then w_1 and w_2 lie in the same tree emanating from C.

Proof. According to the definition of S_{w_1} there exist directed paths $\tilde{P}_1$ and $\tilde{P}_2$ in $\tilde{D}$ leading from y_1 and y_2 to w_1. Clearly, $\tilde{P}_i$ cannot contain u. If w_1 was situated on the cycle C it would be impossible to reach any other `F` tagged node from both y_1 and y_2. Therefore, w_1 lies outside C so that necessarily $C-u$ is a subgraph of S_{w_1}. The paths $\tilde{P}_1$ and $\tilde{P}_2$ start separately but then unite in a common final node z on C before they leave the cycle to enter the emanating tree T that contains w_1. Note that necessarily z is `X` $\overline{\texttt{O}}$ tagged. In order to reach w_1 from z it is required that the `O` tagged partner of z lies in T. This leaves z with no outgoing nodes on C, rendering it the only node on C that can be reached both from y_1 and y_2. Thus, any other `F` tagged node that can be reached from both y_1 and y_2 must lie in T as well. □

Corollary 4.35. *Let G be a unicyclic graph with cracked cycle C. Let u be an* `O` *tagged node of C whose* `X` *tagged partner v does not lie on C and let y_1 and y_2 be the neighbors of u on C.*

If there exists an `F` *tagged node w such that both y_1 and y_2 lie in S_w, then C is even.*

Proof. This follows from the proof of Lemma 4.34 and Lemma 3.10. □

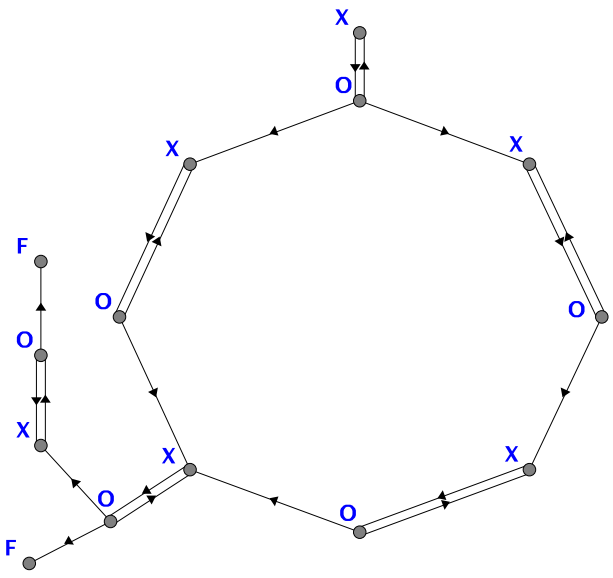

Figure 3: FOX result that illustrates Lemma 4.34

Let us carry our analysis of the structures S_w a little further.

Lemma 4.36. *Let G be a unicyclic graph with cracked cycle C. Let u, u' be two nodes at which C may be cracked. Then there exists a FOX run on G such that the neighbors y_1, y_2 of u within C do not lie in a common subgraph S_w for any* `F` *tagged node w.*

Proof. Since C may be cracked at u and u' it is possible to construct a FOX run on G such that u and u' are the first nodes on C to be tagged, namely `O` tagged. Now consider adjacencies of u and u' in $\tilde{D}$. Any other node of C may at best be an outgoing neighbor of u or u', but never an incoming neighbor. Therefore there cannot exist directed paths in $\tilde{D}$ from y_1 and y_2 leading to the same `F` tagged node w since at least one of these paths would have to run via u or u', which is impossible. □

Lemma 4.37. *Let G be a unicyclic graph with cycle C that can be cracked at node u. Suppose there exists a FOX run on G such that the neighbors y_1 and y_2 of u on C lie in a common S_w for some* `F` *tagged node w.*

Then there exists a subgraph $S_{w'}$ that contains only exactly one of the nodes y_1 and y_2 if and only if C can be cracked at more than one node.

Proof. If there exists a FOX run on G such that the neighbors y_1 and y_2 of u on C lie in a common S_w for some `F` tagged node w, then with respect to Lemma 3.13 this run can be altered such that $w = y_2$. Note that C is necessarily uncut.

Since y_2 has no outgoing neighbors we see that there exists a subgraph $S_{w'}$ that contains only exactly one of the nodes y_1 and y_2 if and only if there is a second `F` tagged node w' in $\tilde{D}$ that lies on a directed path from y_1. Clearly, w' cannot lie on C. With respect to Lemma 3.10 it is clear that a path from y_1 to w' must exit C at an `O` tagged node v since in $\tilde{D}$ the outgoing neighbors of `X` tagged nodes on C all lie on C themselves.

Let T be the tree that is attached to v and contains w'. Assume that in G node v is adjacent to the node z of T. Employ Lemma 3.13 and alter the FOX run such that $\tilde{D}$ remains unchanged for the nodes of $G - T$ but z becomes `F` tagged. Clearly, this run can then be rearranged such that it first operates on the subtree that cracks C at u and after the cracking continues on the subgraph T for as long as possible. But then z cannot become `F` tagged but instead cracks C at v. □

Remark 4.38. *Given a unicyclic graph G with cycle C that may be cracked at a node u, consider a neighbor y of u on C. If y lies in the subgraph S_w of D for some* `F` *tagged node w, then u does not belong to S_w. Now apply a FOX run $\mathcal{A}$ to G to get the final digraph $\tilde{D}^{(\mathcal{A})}$ and let $\tilde{K}^{(\mathcal{A})}$ be the subgraph of $\tilde{D}^{(\mathcal{A})}$ induced by the nodes of the component K of $G - u$ that contains y. From the proof of Theorem 4.26 it follows that it is possible to construct a restriction $\mathcal{B}$ of run $\mathcal{A}$ to K such that the result on K is the same, i.e. the digraphs $\tilde{K}^{(\mathcal{A})}$ and $\tilde{K}^{(\mathcal{B})} = \tilde{D}^{(\mathcal{B})}$ are isomorphic. Therefore, for*

the determination of common subgraphs S_w for the neighbors of u on C as treated in Lemma 4.36 and Lemma 4.37 we may restrict ourselves to K only.

The final case we need to settle is an uncut cracked cycle C whose size is not a multiple of four. Because of Lemma 4.31 we may restrict ourselves to the case $|C| \equiv 2 \mod 4$.

Let us refine the ideas presented in the proof of Theorem 4.26 and assume that C can be cracked at the node u. Let v be the `X` partner of u.

Then a kernel basis of G can be obtained by suitable linear combinations of the vectors of a given basis B' of $\iota(\ker(G-u))$, enforcing the summation rule at node u. We need to characterize under which restrictions on G this strategy admits the construction of a simply structured kernel basis B. In the following, we tacitly require that the basis B' has been obtained by means of Construction 3.14.

Since C is uncut the neighbors y_1 and y_2 of u on C in G are $\overline{\texttt{0}}$ tagged. We will distinguish three types of kernel vectors of $G-u$. Depending on whether none, exactly one or both of y_1 and y_2 are assigned nonzero weights by a given vector $x \in \iota(\ker(G-u))$, we say that x is type τ_0, τ_1 or τ_2, respectively.

Next, we make several simple observations:

Observation 4.39. *Because of $|C| \equiv 2 \mod 4$ a type τ_2 vector always assigns the same values to y_1 and y_2. In the following, we may therefore assume that B' consists of vectors that assign only 0 or 1 to y_1, y_2.*

Observation 4.40. *Consider the component H of $G-u$ that contains v. There exists a kernel vector that does not vanish on v since the restriction of the original FOX run on G to H yields the same node labels, except that v becomes `F` tagged. Therefore, the given basis of $\iota(\ker(G-u))$ always contains a type τ_0 vector.*

Observation 4.41. *Since y_1 is $\overline{\texttt{0}}$ tagged we see that B' always contains at least one type τ_1 or τ_2 vector.*

Observation 4.42. *The basis B' contains a type τ_1 vector if and only if for the given FOX run there exists a subgraph S_w that only contains exactly one of the nodes y_1, y_2. Likewise, it contains a type τ_2 vector if and only if y_1 and y_2 lie in a common S_w.*

In view of the previous observation and Lemma 4.36 we see that for a unicyclic graph whose cycle can be cracked at more than one node we may assume a basis B' that does not contain a type τ_2 vector. On the other hand, it may be assumed that a given basis of $\iota(\ker(G-u))$ only contains at most one type τ_2 vector since the difference of two type τ_2 vectors is of type τ_0.

Consequently, the most simple linear combinations of vectors from B' that lead to vectors which have only entries from $\{0, 1, -1\}$ and satisfy the summation rule at node

u are of the types $\tau_0 - \tau_0$, $\tau_1 - \tau_0$, $\tau_2 - \tau_0 - \tau_0$, and $\tau_2 - \tau_1 - \tau_0$ (multiple vectors of the same type in an expression are assumed to be distinct). With respect to the previous observations and assumptions it is clear that the construction of a simply structured basis of $\ker G$ is possible if and only if it may succeed using only such linear combinations.

Keeping all this in mind we may reason as follows:

Suppose that B' does not contain a type τ_2 vector. Choose a fixed type τ_0 vector and subtract it from the remaining vectors of B'. Then the difference vectors form a simply structured basis of $\ker G$.

If B' contains a type τ_2 vector, then such a construction can succeed if and only if either B' contains a second type τ_0 vector or a type τ_1 vector.

So, in order to solve the final open case we just need to put together the pieces collected so far:

Theorem 4.43. *Let G be a unicyclic graph with uncut cracked cycle C. Let $|C| \equiv 2 \mod 4$. Then G has a simply structured kernel basis if and only if one of the following conditions is satisfied:*

1. *C can be cracked at more than one node.*
2. *There exist at least two nodes that may crack C at the same node.*

Proof. G has a simply structured kernel basis if and only if the construction from the proof of Theorem 4.26 succeeds. Using the results of a suitable FOX run construct a basis B' of $\iota(\ker(G - u))$ as described before. We may assume that B' contains at most one type τ_2 vector.

Then G has a simply structured kernel basis if and only if either B' can be chosen not to contain a type τ_2 vector or to contain a second type τ_0 vector or a type τ_1 vector.

Firstly, according to Lemma 4.36 the occurrence of a type τ_2 vector can be avoided if and only if C can be cracked at more than one node. Secondly, more than one type τ_0 vector exists if and only if there exists a FOX run such that C gets cracked at u, and in D there exist at least two $\overline{0}$ tagged neighbors of u outside C. Equivalently there exist a least two nodes that may crack C at the same node. And finally, Lemma 4.37 states that a type τ_2 vector is accompanied by a τ_1 vector exactly when we could have avoided the type τ_2 vector in the first place. □

We may sum up all cases as follows:

Theorem 4.44. *Let G be a unicyclic graph with cycle C. Then G has a simply structured kernel basis unless $|C| \equiv 2 \mod 4$ and C can be cracked by exactly one node.*

Proof. Collect the results from Theorem 4.23, Theorem 4.25, Corollary 4.29, Corollary 4.31, Theorem 4.33, and Theorem 4.43. □

The criteria we have presented so far in order to achieve a complete characterization of all unicyclic graphs with simply structured kernel bases are formulated in terms of a particular behavior of the FOX algorithm. We now seek to give a purely algebraic version.

Lemma 4.45. *Let G be a unicyclic graph with cycle C. Let u be a node of C and v one of its neighbors outside C. Further, let T be the tree that is connected to C by the edge uv. Then v may crack C at u if and only if there exists a vector $x \in \ker T$ that does not vanish on v.*

Proof. Suppose that C gets cracked at u by v. Restricting the FOX run to the nodes of T only, we obtain a valid FOX run on T such that v is type `F`. According to Lemma 3.9 there exists a vector $x \in \ker T$ that does not vanish on v.

Conversely, by Lemma 3.9 and Lemma 3.13 there exists a FOX run $\mathcal{A}$ on T such that v is type `F`. This run $\mathcal{A}$ represents a partial FOX run $\mathcal{B}'$ on G. Choose v as the next `X` tagged node since it only is connected to u and by construction has no outgoing neighbors among the other nodes of T. But this means that C gets cracked at u by v. □

With the help of Lemma 4.45 we may now rephrase Theorem 4.44 as follows:

Theorem 4.46. *Let G be a unicyclic graph with cycle C. Let T_i be the trees emanating from C and assume that each tree T_i is attached to C by an edge v_iu_i such that u_i and v_i are nodes of C and T_i, respectively.*

Then G has a simply structured kernel basis unless $|C| \equiv 2 \mod 4$ and there exists exactly one index i such that $\ker T_i$ does not completely vanish on v_i.

Having found an algorithmic and an algebraic characterization, let us now derive a purely structural one. Beforehand, we need to elaborate on the connection between the FOX algorithm and maximum matchings on trees. By a matching of a graph we understand a set of independent edges of the graph. A maximum matching comprises the greatest possible number of edges.

Lemma 4.47. *Let M be a maximum matching of the tree T. Then M covers at least one leaf of T.*

Proof. Assume to the contrary that no leaves are covered by M. Let v_1 be such a leaf and v_2 its neighbor. Then v_2 must be covered because M is maximum. Let v_3 be the neighbor of v_2 with $v_2v_3 \in M$. By assumption v_3 cannot be a leaf. Therefore v_3 has another neighbor v_4 with $v_4 \neq v_2$. v_4 cannot be covered by M because otherwise $v_1, \dots, v_4$ would form an augmenting path, contradicting the choice of M. Now the situation for v_4 is like it was for v_2. Continue the argument. Since T is a finite graph, we arrive at a contradiction. □

Lemma 4.48. *Let M be a maximum matching of the tree T. Then there exists a FOX run on T such that M can be obtained by collecting the edges between the* `X` *tagged nodes and their* `O` *tagged partners.*

Proof. Let T_i be a subtree of T and let $M_i = M \cap E(T_i)$ be the restriction of M to T_i. Let the directed graph $\tilde{T}^{(i)}$ be a valid intermediate graph of a FOX run on T. Further, let $\tilde{T}_i$ be the restriction of $\tilde{T}^{(i)}$ to T_i (including labels).

Now we make the following assumptions:

(a) M_i is a maximum matching on T_i.

(b) None of the nodes of $\tilde{T}_i$ have been tagged.

(c) The only nodes of $\tilde{T}^{(i)}$ eligible for `X` tagging lie in the subgraph $\tilde{T}_i$.

(d) Every node eligible for `X` tagging in $\tilde{T}_i$ is also eligible in $\tilde{T}^{(i)}$.

(e) In $\tilde{T}^{(i)}$ the edges between the `X` tagged nodes and their `O` tagged partners form the set $M - M_i$.

(f) None of the untagged nodes of $\tilde{T}^{(i)} - \tilde{T}_i$ are covered by M.

By Lemma 4.47 and assumption (a) there exists a leaf $v_1^{(i)}$ of T_i that is covered by M_i. Let $v_2^{(i)}$ be its neighbor in T_i. Clearly, the edge $e = v_1^{(i)}v_2^{(i)}$ belongs to M_i. We conclude from assumptions (b), (c) and (d) that $v_1^{(i)}$ is eligible for `X` tagging, both in $\tilde{T}_i$ and $\tilde{T}^{(i)}$. In both graphs we can perform a step of the FOX algorithm that tags $v_1^{(i)}$ with `X` and then $v_2^{(i)}$ with `O`. From $\tilde{T}^{(i)}$ we obtain the new intermediate graph $\tilde{T}^{(i+1)}$.

Let $N_i = \{v_3^{(i)}, \dots, v_{s_i}^{(i)}\}$ be the neighbors of $v_2^{(i)}$ besides $v_1^{(i)}$. We can deduce from assumptions (c) and (d) that none of the nodes of N_i have any outgoing edges in $\tilde{T}^{(i+1)}$, hence we already know they will become `F` tagged at the end of the FOX run. The nodes of N_i cannot be covered by M because of assumptions (e) and (f). Now construct the subtree T_{i+1} of T by removing the nodes $v_1^{(i)}, \dots, v_{s_i}^{(i)}$ from T_i. Note that $M_{i+1} = M_i - e$. Incrementing i by one, it is straightforward to see that assumptions (a) to (f) hold again.

Starting with $T_1 = T$ and $M_1 = M$ we see that assumptions (a) to (f) hold for $i = 1$. Iterate the induction step described above until $M_i = \emptyset$. Complete the FOX run on $\tilde{T}^{(i)}$. It is clear by construction that FOX will only assign `F` tags to the remaining untagged nodes, which concludes the proof. □

Lemma 4.49. *Let M be the set of edges between the* `X` *tagged nodes and their* `O` *tagged partners obtained by a FOX run on a given tree T. Then M is a maximum matching on T.*

Proof. Obviously M is a matching on T. Exactly the `F` tagged nodes are not covered by M. It is proven in [6] that the rank of the adjacency matrix of a tree equals twice the cardinality of a maximum matching. In view of Theorem 3.15 this means that indeed M is a maximum matching. □

Corollary 4.50. *Exactly the* `O` *tagged nodes of a tree are covered by every single maximum matching.*

Proof. This follows from Lemmas 3.11, 3.13, 4.48, and 4.49. □

Now we can finally give the desired structural characterization:

Theorem 4.51. *Let G be a unicyclic graph with cycle C. Let T_i be the trees emanating from C and assume that each tree T_i is attached to C by an edge v_iu_i such that u_i and v_i are nodes of C and T_i, respectively.*

Then G has a simply structured kernel basis unless $|C| \equiv 2 \mod 4$ and there exists exactly one index i such that v_i is not covered by some maximum matching of T_i.

Proof. Rewrite Theorem 4.46 using Corollary 4.50 and Lemma 3.9. □

5. Conclusion

It can be seen from Figure 1 that not every unicyclic graph has a simply structured kernel basis. This is different from the situation for trees, cf. Theorem 3.15 or [10]. It becomes obvious from the previous section that a complete characterization of all unicyclic graphs that admit simply structured kernel bases requires some effort. On the other hand, our findings reveal interesting properties of the FOX algorithm and the kernels of unicyclic graphs.

We have achieved the desired characterization and proven an algorithmic, an algebraic and a structural version. Except for a very special case we find that almost every unicyclic graphs admits a simply structured kernel basis. We see from the proofs of our findings that it is possible to derive such a basis by an initial FOX run on the unicyclic graph, employment Construction 3.14 for certain trees and finally suitable embedding and linear combination.

The discovery of a general characterization that states exactly which graphs admit simply structured kernel bases is still an open issue and requires further research.

References

[1] S. Akbari, A. Alipour, E. Ghorbani and G. Khosrovshahi, $\{-1, 0, 1\}$-basis for the null space of a forest, *Linear Algebra Appl.,* **414**(2006), 506 - 511.

[2] S. Akbari, N. Ghareghani, G.B. Khosrovshahi and H.R. Maimani, The kernels of the incidence matrices of graphs revisited, *Linear Algebra Appl.,* **414**(2-3) (2006), 617 - 625.

[3] N. Biggs, *Algebraic graph theory*, Second ed., Cambridge Mathematical Library, Cambridge University Press, Cambridge, 1993.

[4] D.M. Cvetković and I.M. Gutman, The algebraic multiplicity of the number zero in the spectrum of a bipartite graph, *Mat. Vesnik,* **9**(1972), 141 - 150.

[5] S. Fiorini, I. Gutman and I. Sciriha, Trees with maximum nullity, *Linear Algebra Appl.,* **397**(2005), 245 - 251.

[6] G.H. Fricke, S.T. Hedetniemi, D.P. Jacobs and V. Trevisan, Reducing the adjacency matrix of a tree, *Electron. J. Linear Algebra,* **1** (1996), 34 - 43.

[7] C. Godsil and G. Royle, *Algebraic graph theory,* Vol. 207 of Graduate Texts in Mathematics, Springer-Verlag, New York, 2001.

[8] F. Hazama, On the kernels of the incidence matrices of graphs, *Discrete Math.,* **254**(2002), 165 - 174.

[9] P.E. John and G. Schild, Calculating the characteristic polynomial and the eigenvectors of a tree, *Match,* **34** (1996), 217 - 237.

[10] J.W. Sander and T. Sander, On Simply structured bases of tree kernels, *AKCE J. Graphs. Combin.,* **2**(2005), 45 - 56.

[11] T. Sander, On certain eigenspaces of cographs, (To appear).

[12] I. Sciriha and I. Gutman, Nut graphs: Maximally extending cores, *Utilitas Math.,* **54**(1998), 257 - 272.

[13] I. Sciriha and I. Gutman, Minimal configurations and interlacing, *Graph Theory Notes of New York,* **49** (2005), 38 - 40.

[14] R.H. Villarreal, Rees algebras of edge ideals, *Commun. Algebra,* **39**(1995), 3513 - 3524.