Applicable Analysis and Discrete Mathematics
available online at http://pefmath.etf.rs

Appl. Anal. Discrete Math. **14** (2020), 122–137.
https://doi.org/10.2298/AADM180331003I

# RECOGNIZING GENERALIZED SIERPIŃSKI GRAPHS

*Wilfried Imrich and Iztok Peterin**

Let $H$ be an arbitrary graph with vertex set $V(H) = [n_H] = \{1, \ldots, n_H\}$. The *generalized Sierpiński graph* $S_H^n$, $n \in \mathbb{N}$, is defined on the vertex set $[n_H]^n$, two different vertices $u = u_n \ldots u_1$ and $v = v_n \ldots v_1$ being adjacent if there exists an $h \in [n]$ such that $(a)$ $u_t = v_t$, for $t > h$, $(b)$ $u_h \neq v_h$ and $u_h v_h \in E(H)$, and $(c)$ $u_t = v_h$ and $v_t = u_h$ for $t < h$. If $H$ is the complete graph $K_k$, then we speak of the Sierpiński graph $S_k^n$. We present an algorithm that recognizes Sierpiński graphs $S_k^n$ in $O(|V(S_k^n)|^{1+1/n}) = O(|E(S_k^n)|)$ time. For generalized Sierpiński graphs $S_H^n$ we present a polynomial time algorithm for the case when $H$ belong to a certain well defined class of graphs. We also describe how to derive the base graph $H$ from an arbitrarily given $S_H^n$.

## 1. INTRODUCTION

Sierpiński graphs $S_k^n$ were introduced and studied for the first time by Klavžar and Milutinović in [**17**]. The study was motivated in part by the fact that for $k = 3$ these graphs are isomorphic to the Tower of Hanoi graphs [**9**], and in part by topological studies. For details about the latter motivation see Lipscomb's book [**19**]. Since the introductory paper in 1997 Sierpiński graphs have become quite popular, compare the recent survey [**13**] with 121 references, or the recent monograph [**12**]. In particular, let us mention that the metric properties of Sierpiński graphs were studied already in [**17**] and later in [**10, 11, 14, 22**], which will be of some relevance here.

Sierpiński graphs $S_k^n$ represent, roughly speaking, graphs with a fractal structure, where the base graph is represented by the complete graph $K_k$. The natural

*Corresponding author. Iztok Peterin
2010 Mathematics Subject Classification. 05C85, 68R10.
Keywords and Phrases. Sierpiński graphs; generalized Sierpiński graphs; algorithm.

idea is to replace $K_k$ by an arbitrary graph $H$, called base graph. This idea was first presented by Gravier et al. in [**8**]. In recent years several publications followed this idea, and it seems that generalized Sierpiński graphs will become similarly popular as Sierpiński graphs: total colorings of generalized Sierpiński graphs were investigated in [**7**], the Randić index in [**24**], and the generalized Randić index in [**5**]. The strong metric dimension was treated in [**3**], and in [**4**] the chromatic number, the vertex cover number, the clique number and the domination number of generalized Sierpiński graphs. A discussion on Roman domination can be found in [**23**], and in [**18**] an investigation of generalized Sierpiński graphs with respect to connectivity and other properties, such as Hamiltonicity. Finally, distances in generalized Sierpiński graphs are discussed in [**6**].

Interestingly, generalized Sierpiński graphs have not been investigated from the recognition point of view, that is, from the point of view whether there exists an effective algorithm that decides whether a given graph is isomorphic to a Sierpiński graph, or to a generalized Sierpiński graph. We partially fill this gap. In next section we present a detailed definition of generalized Sierpiński graphs, together with basic properties. In Section 3 we continue with an algorithm for the recognition of Sierpiński graphs that is almost linear in the number of vertices, but linear in the number of edges of $S_k^n$. Then we discuss the general case and present a polynomial algorithm for the recognition of generalized Sierpiński graphs with a base graph $H$, where each edge of $H$ is contained in a triangle and where we can check whether a given graph is isomorphic to $H$ in polynomial time. In the last section this is complemented with a polynomial algorithm for the general case that computes a candidate for the base graph.

## 2. GENERALIZED SIERPIŃSKI GRAPHS

Let $H$ be an arbitrary graph with vertex set $V(H) = [n_H] = \{1, \ldots, n_H\}$. The *generalized Sierpiński graph* $S_H^n$, $n \in \mathbb{N}$, is defined on the vertex set $[n_H]^n$, two vertices $u = u_n \ldots u_1$ and $v = v_n \ldots v_1$ being adjacent if there exists an $h \in [n]$ such that

- $u_t = v_t$, for $t > h$,
- $u_h \neq v_h$ and $u_h v_h \in E(H)$, and
- $u_t = v_h$ and $v_t = u_h$ for $t < h$.

We say that $H$ is the *base graph* of $S_H^n$. Clearly $H \cong S_H^1$, which means that every graph is a generalized Sierpiński graph. Therefore we are interested only in the case when $n > 1$. If $H \cong K_k$, then we obtain the Sierpiński graph $S_k^n \cong S_{K_k}^n$. See Figure 1 for the generalized Sierpiński graph $S_H^2$, where $H$ is a house graph, and the Sierpiński graph $S_3^3$. The edge set can be written in compact form as

$$E(S_H^n) = \{\underline{s}ij^{h-1}\underline{s}ji^{h-1} : h \in [n], \underline{s} \in [n_H]^{n-h}, ij \in E(H)\}.$$

It is easy to see that $|V(S_H^n)| = n_H^n$, $\Delta(S_H^n) = \Delta(H) + 1$ and $\delta(S_H^n) = \delta(H)$. In particular, $\delta_{S_H^n}(i^n) = \delta_H(i)$ and $\delta_{S_H^n}(i^{n-1}j) = \delta_H(j)+1$, where $ij \in E(H)$ and $\delta(v)$ represents the degree of the vertex $v$. If we set $q_n = |E(S_H^n)|$, then $q_n = n_H q_{n-1}+q_1$, where $q_1 = |E(H)|$. By a simple computation we obtain

$$|E(S_H^n)| = |E(H)|\frac{n_H^n - 1}{n_H - 1}. \tag{1}$$

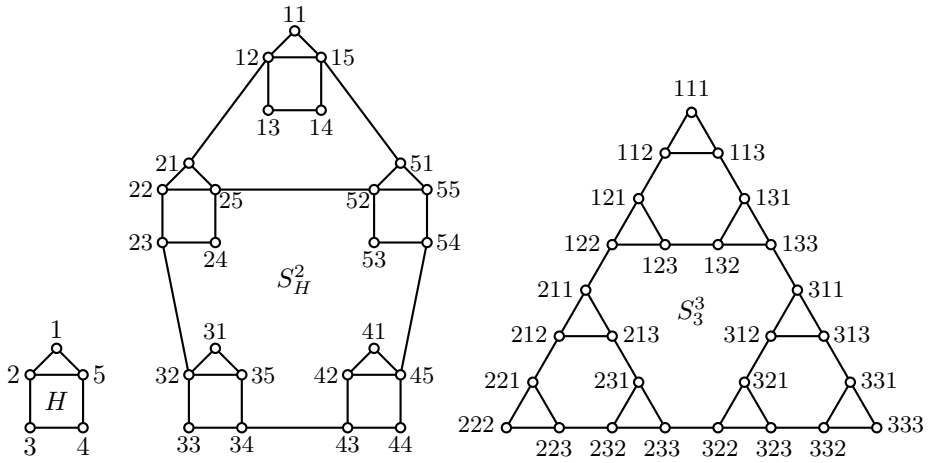


Figure 1: The house graph $H$, $S_H^2$ and $S_3^3$.

A vertex of the form $i^n$, $i \in [n_H]$, of $S_H^n$ is called an *extreme vertex*. Note that $S_H^n$ contains $n_H$ extreme vertices. If $n \geq 2$, then for $i \in [n_H]$, let $iS_H^{n-1}$ be the subgraph of $S_H^n$ induced by the vertices of the form $iv_{n-1} \dots v_1$. More generally, for given $\underline{s} \in [n_H]^r$, we denote by $\underline{s}S_H^{n-r}$ the subgraph of $S_H^n$ induced by the vertices of the form $\underline{s}v_{n-r} \dots v_1$. Note that $iS_H^{n-1}$ is isomorphic to $S_H^{n-1}$, and, more generally, $\underline{s}S_H^{n-r}$ is isomorphic to $S_H^{n-r}$. We use the notation $H_u$ for the subgraph $\underline{s}S_H^1$, $\underline{s} \in [n_H]^{n-1}$ of $S_H^n$ (which is isomorphic to $H$) that contains the vertex $u$.

An edge of $S_H^n$ between $u_n u_{n-1} \dots u_2 i$ and $u_n u_{n-1} \dots u_2 j$, $i \neq j$, will be called a *base edge*. Clearly, in such a case $ij \in E(H)$. Such a base edge is contained in a unique subgraph $u_n u_{n-1} \dots u_2 S_H^1 \cong H$ of $S_H^n$. The other edges will be called *non-base edges*. For $ij \in E(H)$ the edge between $ij^{n-1}$ and $ji^{n-1}$ is the unique edge between $iS_H^{n-1}$ and $jS_H^{n-1}$. Also for $j\ell \in E(H)$ there exists an edge between $\underline{s}j\ell^{n-r-1}$ and $\underline{s}\ell j^{n-r-1}$ for some $\underline{s} \in [n_H]^r$. Recall that in the case of Sierpiński graphs there is an edge between any two vertices of $H$, because $H \cong K_k$.

A vertex of the form $\underline{s}i^p$, $\underline{s} \in [n_H]^{n-p}$, $i \in [n_H]$ and $p \geq 2$, is called *$p$-extreme*. The reason for this is that it is an extreme vertex in a subgraph $\underline{s}S_H^p$ of $S_H^n$. Since it is an extreme vertex in $\underline{s}S_H^p$ all edges in $\underline{s}S_H^p$ incident with $\underline{s}i^p$ are base edges.

On the other hand, there may exist non-base edges incident with $\underline{s}i^p$. Such a non-base edge exists whenever the last label $s_{n-p}$ from $\underline{s}$ is adjacent to vertex $i$ in $H$. The other vertex of such a non-base edge is of the form $s_n \dots s_{n-p+1} i s_{n-p}^p$, where $\underline{s} = s_n \dots s_{n-p+1} s_{n-p}$. In Section 4 we will need a special kind of 2-extreme vertices. A 2-extreme vertex $u$ of $S_H^n$ is called a *proper 2-extreme vertex* if $\delta_H(u) = \delta_{S_H^n}(u)$. Proper 2-extreme vertices are 2-extreme vertices which are not incident with a non-base edge.

## 3. RECOGNIZING SIERPIŃSKI GRAPHS $S_k^n$

In this section we present a recognition algorithm that checks whether a given graph $G$ is a Sierpiński graph $S_k^n$. The ideas represented here will be useful for generalized Sierpiński graphs too. The idea of the algorithm is to start in an extreme vertex $v$ and proceed in BFS order. As usual in BFS algorithms it partitions the vertices of $G$ into distance levels $\ell_i(v) = \{u \in V(G) : d_G(v,u) = i\}$ and, if $u \in \ell_i(v)$, then we set $\ell(u) = i$. With respect to these levels we say that an edge $uw$, $\ell(u) = i$, is a *down*, *up* or *cross* edge with respect to $u$ if $\ell(w) = i+1$, $\ell(w) = i-1$ or $\ell(w) = i$, respectively. The next lemma, from [**17**], shows how distances to extreme vertices in $S_k^n$ can be computed. Setting

$$\rho_{i,j} = \begin{cases} 1 & : \quad i \neq j\,, \\ 0 & : \quad i = j\,, \end{cases}$$

the following holds for Sierpiński graphs.

**Lemma 1.** [**17**] *Let $u_n \dots u_1$ and $i^n$ be vertices of $S_k^n$. Then*

$$d_{S_k^n}(u_n \dots u_1, i^n) = \rho_{u_n,i}\rho_{u_{n-1},i} \cdots \rho_{u_1,i}\,,$$

*where the right-hand side is a binary number with digits $\rho_{u_j,i}$. Moreover, any shortest path between $u_n \dots u_1$ and $i^n$ is unique.*

A particular case of (1) for Sierpiński graphs is

$$|E(S_k^n)| = \frac{k(k^n-1)}{2}. \tag{2}$$

We will use the following simple lemma.

**Lemma 2.** *Let $v$ be an extreme vertex of $S_k^n$ and the starting vertex of the BFS algorithm. If $d_{S_k^n}(u,v) = 2t$ for some vertex $u \in V(S_k^n)$, then the down neighbors of $u$ induce $K_{k-1}$.*

*Proof.* Without loss of generality we may assume that $v = 1^n$. Let $d_{S_k^n}(u,v) = 2t$. Lemma 1 implies that $u_1 = 1$. Assume that $u = \underline{s}1^q$ for some $\underline{s} \in [k]^{n-q}$. The

subgraph $\underline{s}S_k^q$ is isomorphic to $S_k^q$ and $u$ is the closest to $v$ among all vertices of $\underline{s}S_k^q$ by Lemma 1. Moreover, $u$ is an extreme vertex of $S_k^q$ and its down neighbors induce $K_{k-1}$. This completes the proof. □

A graph $H$ is a *minor* of $G$ if it can be obtain by deleting vertices and edges and by contracting some edges of $G$. The following lemma mirrors the fractal structure of Sierpiński graphs.

**Lemma 3.** *If we contract all base edges in $S_k^n$, then we obtain $S_k^{n-1}$.*

*Proof.* All base edges of $S_k^n$ are in complete subgraphs $K_k$ and every edge from any complete graph $K_k$ is a base edge. Every complete subgraph $K_k$ of $S_k^n$ is of the form $\underline{s}S_k^1$ for some $\underline{s} \in [k]^{n-1}$. By contraction of edges in $\underline{s}S_k^1$ we obtain a new vertex $\underline{s}$. It is clear that the remaining graph is isomorphic to $S_k^{n-1}$. □

It is not surprising that $S_k^{n-1}$ is a minor of $S_k^n$, because it is its subgraph. But the contractions mentioned in Lemma 3 are the cornerstone of our algorithm. Lemma 3 means that for a given graph we have to repeat the same loop $n-1$ times to end with a $K_k$ if $G$ is a Sierpiński graph. Also, it is easy to obtain $k$. It is the number of vertices in $N[v]$ where $v$ is a vertex of minimum degree in $G$. On the other hand, $k$ is equal to the number of vertices of minimum degree, which seems even more handy for our purpose. Now one can compute $n = \log_k |V(G)|$. It only remains to check at each step whether the down neighbors of vertices of even distance from extreme vertices induce $K_{k-1}$ because of Lemma 2.

**Algorithm** 1
**Input:** A bi-regular graph $G$ with $k$ vertices of minimum degree $\delta$ and $k^n - k$ vertices of degree $\delta + 1$, and a vertex $v$ of minimum degree.
**Output:** YES if $G \cong S_k^n$ and NO otherwise.

**Begin**

1. **for** $i = 1$ **to** $n$ **do**

   1.1. **if** there exists a vertex of even distance from $v$ such that its down neighbors do not induce $K_{k-1}$ **then** NO and STOP;

   1.2. contract all base edges;

2. **if** $G = K_1$ **then** YES; **else** NO

**End**

**Theorem 4.** *Algorithm 1 correctly recognizes Sierpiński graphs $S_k^n$ within $O(k \cdot k^n) = O(|V(G)|^{1+1/n}) = O(|E(G)|)$ time.*

*Proof.* The correctness of Algorithm 1 can be shown by induction on $n$. If $n = 1$, then $S_k^1 \cong K_k$ by the choice of $v$. Clearly the down neighbors of $v$ induce $K_{k-1}$ and the contraction of all edges results in a $K_1$. Now let $n > 1$ and suppose that

Algorithm 1 correctly recognize $S_k^{n-1}$. If it stops at $i = 1$, then $G$ is not a Sierpiński graph by Lemma 2. Otherwise, if the algorithm does not stop for $i = 1$, then all down neighbors of every vertex of even distance from $v$ induce $K_{k-1}$. If $G \cong S_k^n$, then after contraction of all base edges only $S_k^{n-1}$ remains by Lemma 3, and then the output is YES by the induction hypothesis. Similarly, if we did not get $S_k^{n-1}$ after contraction of all base edges, then $S_k^n$ is not a Sierpiński graph by the same lemma, and the output is NO by the induction hypothesis. Thus Algorithm 1 correctly recognize Sierpiński graphs.

It is clear that we can initiate the algorithm (check for the degrees and the number of vertices) in the prescribed time complexity. In each loop we have to consider every edge a constant number of times as we proceed in BFS order. Indeed, to check whether the down neighbors induce a complete graph, one only needs to count if there are exactly $\frac{(k-1)(k-2)}{2}$ cross edges among them. This is easy because of the fact that every vertex in the $K_k$ can have at most one neighbor outside of the $K_k$. If not, then we end the algorithm. Moreover, in each loop the number of edges decreases. Hence with respect to (2), we all together count

$$O\left(\sum_{i=1}^{n} |E(S_k^i)|\right) = O\left(\sum_{i=1}^{n} \frac{k(k^i-1)}{2}\right) = O\left(\frac{k}{2}\sum_{i=1}^{n}(k^i-1)\right) = O(k^{n+1})$$

operations. Because $|V(G)| = k^n$, $k = \sqrt[n]{|V(G)|}$, and by (2), we end with the desired time complexity $O(k|V(G)|) = O(|V(G)|^{1+1/n}) = O(|E(G)|)$. □

## 4. RECOGNIZING GENERALIZED SIERPIŃSKI GRAPHS FOR A SPECIAL CLASS OF BASE GRAPHS

As we will see, the most difficult part in recognizing generalized Sierpiński graphs is the task to distinguish base edges from non-base edges. In particular, this causes severe problems in the case of 2-extreme vertices, which are not proper (when $n > 2$). We are not able to do this efficiently in general. We can solve this in reasonable time for the following base graphs. A graph $H$ is called an *edge-triangle graph*, or ET graph for short, if each edge of $H$ is contained in a triangle. Clearly trees are not ET graphs. On the other hand, two-connected chordal graphs are ET graphs. Let $G$ be a graph. We can construct an ET graph $H$ from $G$ as follows. Let $V(H) = V(G) \cup E(G)$ and $E(H) = E(G) \cup \{(eu, ev) : e \in E(G), e = uv\}$. In other words, for every edge $e = uv$ of $G$ add a new vertex $e$ in $H$ and two new edges $eu$ and $ev$ in $H$. In particular, the *sun* is a graph obtained by this construction when we start with a cycle $G = C_n$.

In Section 4 we will present an algorithm that obtains the base graph $H$ of a generalized Sierpiński graph. However, in the case of an ET graph being a base graph, we can use a different, faster approach based on the following lemma. The lemma itself holds for an arbitrary graph $H$, but can be used efficiently only for ET graphs as explained later. The proof is straightforward, because two vertices of $H_u$

that have a neighbor outside of $H_u$ are at distance at most two, and is therefore omitted.

**Lemma 5.** *Let $H$ be a graph, $u \in V(S_H^n)$, $n \geq 2$, and $e = xy$, $x \in V(H_u)$, is a non-base edge. If we start the BFS algorithm in $u$, then $y$ is incident with a unique up edge and no cross edges.*

Two levels of the BFS algorithm are enough to recognize if a vertex $u$ from $S_H^n$ is a proper 2-extreme vertex. To describe this it is convenient to define the concept of an $\ell_2$-*ancestor*. Let $v$ be a vertex of distance at least 3 from $u$. Then the ancestor of $v$ in $\ell_2(u)$ with respect to the BFS tree is called an $\ell_2$-ancestor of $v$ and denoted by $a_2(v)$. For $v \in \ell_2(u)$ we set $a_2(v) = v$. With this terminology we can say that if we start the BFS algorithm in a proper 2-extreme vertex $u$, then every $\ell_2$-ancestor of a vertex that is not in $H_u$, has exactly one up edge and no cross edge. On the other hand every $\ell_2$-ancestor with respect to $u$ from $H_u$ has at least two up edges or at least one cross edge as $H$ is an ET graph.

With Lemma 5 we can use a vertex $u$ of maximum degree in $S_H^n$ to find $H_u$. Indeed, any vertex of maximum degree is incident with a non-base edge. Moreover, all vertices from $H_u$ that are incident to a non-base edge are at distance at most two to $u$. Hence only three levels of the BFS algorithm are enough to detect all non-base edges surrounding $H_u$. In addition one also has to check that every vertex from $H_u$ has at most one neighbor outside of $H_u$. To obtain the whole $H_u$ it is enough to find a connected component that contains $u$ once the mentioned non-base edges are deleted.

From this we obtain $n = \log_{|V(H)|} |V(G)|$ and, if $n$ is not a natural number, then $G$ is not a generalized Sierpiński graph. However we need to find all $n_H^{n-1}$ copies of base graph $H$ if $G \cong S_H^n$ as well as the 2-extreme vertex of $H_u$. For this we can use the next lemma.

**Lemma 6.** *Let $H$ be an ET graph, $u$ be a 2-extreme vertex of $S_H^n$, which is ordered by a BFS algorithm with base $u$, and $v \notin V(H_u)$ is a vertex adjacent to a vertex $x$ in $H_u$. If $u$ and $x$ are in a common clique $C_t$, $t \geq 3$, of $H_u$, then there exists a clique $C_t'$ such that all of its vertices, with the exception of at most two, have cross neighbors which are not in $H_v$. Moreover, one exception is $v$ and the other exception is the 2-extreme vertex of $H_v$, and these two vertices are the same when $u = x$ and $u$ is a not proper 2-extreme vertex.*

*Proof.* First, let $u = x$. Then $u$ is a 2-extreme vertex which is not proper, $n > 2$, and $u = x$ is incident with a non-base edge $e = uv$, where $v$ is also a 2-extreme vertex that is not proper. By Lemma 5 $v$ has no cross edges and as $H$ is an ET graph every down neighbor of $v$ has a cross neighbor which is not in $H_v$.

Suppose now that $u \neq x$. Clearly $v$ is two distance levels higher than $u$. Let $w \neq v$ also be two levels above $u$ in BFS order, where $w$ is not in $H_u$, such that the parents $p(w) \neq p(v)$ are in clique $C_t, t \geq 3$. Let $\phi_v$ be an isomorphism between $H_u$ and $H_v$ such that $\phi_v(u) = v$, and let $\phi_w$ be an isomorphism between $H_u$ and $H_w$, where $\phi_w(u) = w$. It is easy to see that $C : p(v)v\phi_v(p(w))\phi_w(p(v))wp(w)$ is a six

cycle. Notice that $\phi_v(p(w))$ is in a clique $C'_t$ as $\phi_v$ is an isomorphism. Therefore, $\phi_v(p(w))$ has a cross neighbor $\phi_w(p(v))$ in $H_w$, which is outside of $H_v$ but is in $S^2_H$ that contains $u$. Since $w$ was chosen such that its parent is in $C_t$, and since $p(w) \neq p(v)$, all vertices of $C'_t$ have cross neighbors not in $H_v$ with the exception of $v$ and $\phi_v(p(v))$. The latter is therefore a 2-extreme vertex of $H_v$ and we can distinguish $\phi_v(p(v))$ from other vertices in $C'_t$. □

Notice that we can distinguish algorithmically in above proof if a vertex has a cross neighbor outside of $H_v$. This is indeed so, since such cross neighbors have different $\ell_2$-ancestors.

We also need the 2-extreme vertex of $H_u$. Lemma 6 comes handy also for that. If all vertices of $H_u$ that are incident with a non-base edge form an open neighborhood of some vertex $z$, then $z$ is the 2-extreme vertex of $H_u$. (Notice that there may exist more than one such vertex, but then they have the same neighborhood and may replace each other.) Otherwise one of the vertices of $H_u$ incident with a non-base edge must be an improper 2-extreme vertex $z$. In such a case the closed neighborhood of $z$ in $H_u$ contains exactly the vertices of $H_u$ incident with non-base edges. We claim that if $z \neq u$, then there exists exactly one $\ell_2$-ancestor $x$ such that its children have no up neighbors with different $\ell_2$-ancestors. Clearly $x$ is such that $z = p(x)$. To see that others have up neighbors with different $\ell_2$-ancestors we involve the cycle $C$ from the proof of Lemma 6 with slightly different notation. Here we have $C : uu'\phi_{u'}(w)\phi_{w'}(u)w'w$ where $w$ is adjacent to both $u$ and $z$ (which exists for ET graphs), further $uu'$ and $ww'$ are non-base edges and $\phi_t : H_u \to H_t$ for $t \in \{u', w'\}$ is an isomorphism. Clearly $\phi_{w'}(u)$ has two up neighbors $w'$ and $\phi_{u'}(w)$ where $a_2(w') = w' \neq \phi_{u'}(w) = a_2(\phi_{u'}(w))$.

Next we present an algorithm that recognizes generalized Sierpiński graphs $S^n_H$ where $H$ is an ET graph and $n \geq 2$. Not to exceed the length of the algorithm, we present some steps in dense writing style.

For Step 5 we also need the following lemma. It generalises Lemma 3. The proof is very similar, but short.

**Lemma 7.** *If we contract all base edges in $S^n_H$, then we obtain $S^{n-1}_H$.*

*Proof.* Every base graph $H$ of $S^n_H$ is of the form $\underline{s}S^1_H$ for some $\underline{s} \in [k]^{n-1}$. After contraction of the edges in $\underline{s}S^1_H$ we denote the new vertex by $\underline{s}$. It is clear that the remaining graph is isomorphic to $S^{n-1}_H$ as $s_n \dots s_2 s_1$ is adjacent to $s_n \dots s_1 s_2$ in $S^n_H$ whenever $s_1 s_2 \in E(H)$. Therefore $s_n \dots s_3 s_2$ is adjacent to $s_n \dots s_3 s_1$ in $S^{n-1}_H$. □

**Algorithm** 2
**Input:** A connected graph $G$ and a vertex $u \in V(G)$ of maximum degree.
**Output:** YES if $G \cong S^n_H$ for an ET graph $H$ and $n \geq 2$ and NO otherwise.

**Begin**

1. find $H_u$ as described after Lemma 5 and put all vertices not in $H_u$ but adjacent to a vertex in $H_u$ in set $S_u$; in addition add a set $S$ that contains $S_u$; **if** a

vertex from $H_u$ is incident to more than one non-base edge, **then** NO and STOP;

2. **if** $n = \log_{|V(H)|} |V(G)|$ is not a natural number, **then** NO and STOP;

3. find the 2-extreme vertex of $H_u$ as described after Lemma 6; **if** such a 2-extreme vertex of $H_u$ does not exists, **then** NO and STOP;

4. **until** there is a nonempty set in $S$ **do**

   4.1. for every vertex $v$ from $S_u$ find the 2-extreme vertex of $H_v$ with respect to Lemma 6 and delete $v$ from $S_u$. Let $x$ be the 2-extreme vertex of $H_v$; **if** there is no such vertex, **then** NO and STOP;

   4.2. find $H_x$ as described after Lemma 5 and put all vertices, which were not considered yet and are not in $H_x$, but adjacent to a vertex in $H_x$, into the subset $S_x$ and store $S_x$ in $S$;

   4.3. check if $H_u$ and $H_x$ are isomorphic with additional constrain that $u$ maps into $v$ and $p(v)$ maps in $x$; **if** they are not isomorphic, **then** NO and STOP;

5. contract all base edges of every $H_x$, where $x$ is a 2-extreme vertex, in to a vertex called $x$;

6. **repeat** steps 1, 2, 4 and 5 $n - 2$ times with one simplification: now the vertices are already labeled and one can check for an isomorphism between $H_x$ and $H_y$ faster;

7. **if** we end up with a graph isomorphic to an ET graph $H$, **then** YES; **else** NO.

**End**

Before we prove the correctness of Algorithm 2, let us mention that according to Step 4.3. (isomorphism checking) we cannot expect polynomial time complexity for all ET graphs, despite the fact that the number of vertices $n_H$ in $H$ is much smaller than the number of vertices $n_G = n_H^n$ of $G$. Nevertheless, there are several graph classes for which a polynomial time algorithm is known for the solution of the isomorphism problem. Examples for such graphs are trees [**16**], planar graphs [**15**], interval graphs [**2**], circulant graphs [**21**], graphs with bounded treewidth [**1**], graphs of bounded degree [**20**] and others. We say that graph $H$ is in class, $\mathcal{IP}$ if there exists a polynomial algorithm for the isomorphism problem for $H$.

**Theorem 8.** *Algorithm 2 correctly recognize generalized Sierpiński graphs $S_H^n$, where $H$ is an ET graph and $n \geq 2$. If $H$ is in $\mathcal{IP}$, then it runs in polynomial time.*

*Proof.* Let $G$ be an arbitrary connected graph. We will proof the correctness of Algorithm 2 by induction on $n$. The proof of the validity of the induction base (for $n = 2$) and inductive step are very similar. Therefore we explain the inductive step in detail and give some additional remarks for the base. Clearly Step 1 yields the base graph $H$ if it exists by Lemma 5. Notice that $H$ at this stage may not be an ET graph.

Step 2 is clear once $H_u$ is known. For Step 3 it is easy to see that if there exists a vertex $w$ whose open neighborhood contains only vertices of $H_u$ incident with non-base edges, then $w$ is a proper 2-extreme vertex of $H_u$. Otherwise one vertex from $N_{H_u}(w)$ would be the 2-extreme vertex of $H_u$, in contradiction with $w$ not being incident with a non-base edge. (This occur always when $n = 2$ and all 2-extreme vertices of $S_H^2$ are extreme vertices.) So we may assume that there exists no vertex whose open neighborhood satisfies the above condition. (Only possible in the inductive step when $n > 2$.) Therefore the 2-extreme vertex $w$ of $H_u$ is not proper and its closed neighborhood contains only vertices of $H_u$ incident with non-base edges. If $w = u$, then for adjacent vertices $x, y \in N_G(w)$ there exists a six-cycle with three non-base edges and three base edges. Clearly such an edge $xy$ exists as we are interested only in ET base graphs $H$. Then one non-base edge of the six-cycle is incident with $x$, the other with $y$, and the third is a cross edge with respect to $w$ in level 3 of the BFS order. Thus, if we start a BFS algorithm in say $x$, then this six-cycle yields an $\ell_2$-ancestor with a child that has an up edge with a different $\ell_2$-ancestor. The only vertex in $N_G(x)$ where this does not occur, is the 2-extreme vertex which is not proper. Hence $H$ passes Step 3.

In Step 4 we basically traverse the graph from one subgraph $H_u$ to another subgraph $H_x$ adjacent by a non-base edge in a kind of BFS order (Steps 4.1 and 4.2). They must be isomorphic and this is checked in Step 4.3. (For this step we recommend the program NAUTY, which is available at web page https://www3.cs.stonybrook.edu/ algorith/implement/nauty/implement.shtml.) In particular, if $H$ is not an ET graph, then the algorithm will stop at some point in Step 4.1 when we have an edge that is not in triangle, unless this edge is a cross edge in our BFS order where its endvertices have no down neighbors. Also, the algorithm stops if two adjacent subgraphs $H_x$ and $H_y$ are not isomorphic.

After Step 5 we obtain a graph $G'$ on $n_h^{n-1}$ vertices and if $G$ was isomorphic to $S_H^n$ then $G'$ is isomorphic to $S_H^{n-1}$ by Lemma 7. If $n = 2$, then we need to check if $G' \cong H$. This is done in Step 7. If they are isomorphic, then $G \cong S_H^2$ which ends the base of the induction. Otherwise, when $n > 2$, Algorithm 2 correctly recognizes whether $G'$ is isomorphic to $S_H^{n-1}$ for an ET base graph $H$ and $n \geq 2$ by induction hypothesis. Hence Algorithm 2 correctly recognize generalized Sierpiński graphs $S_H^n$ for ET base graphs $H$ and $n \geq 2$.

Now to the time complexity. Clearly each step of Algorithm 2 runs in polynomial time for all steps but Step 4.3. We explain this for Step 2, the other cases are similar. In this step we observe two cases. If $H_u$ contains a proper 2-extreme vertex, then we can find it in $O(\Delta^2(G))$ time. Otherwise, we can check the up edges of the vertices of level $\ell_3$ and find the 2-extreme vertex in time faster than

$O(|E(G)|)$. Hence Algorithm 2 runs in polynomial time if the base graph $H$ is in $\mathcal{IP}$. □

## 5. FINDING A BASE GRAPH OF $S_k^n$

In contrast to Sierpiński graphs one can immediately see some obstacles which make it hard to recognize generalized Sierpiński graphs for a general base graph $H$. First, we do not have a bi-regular graph in general and therefore we cannot get the base graph $H$ of $S_H^n$ just by observing degrees. In the sequel we will solve this by some distance properties of $S_H^n$ for any graph $H$ (see Lemmas 10 and 11). The second issue is that, if we obtain a base graph $H$, then we need to check whether all base graphs are isomorphic. As already mentioned it is well known that until now there exists no polynomial time algorithm for the solution of the graph isomorphism problem. The third problem is that a base graph can have several vertices of minimum degree, which means that we do not know where to start, that is, we do not know which vertex is a potential extreme vertex. (We cannot start in a vertex of maximum degree as in Algorithm 2.) This can be solved for a general base graph $H$, but it takes more time than in case of Sierpiński graphs (see Lemma 9).

The most problematic difference is that we cannot distinguish between base and non-base edges, which was automatic in Sierpiński graphs (as a consequence of Lemma 1).

In the sequel the only similarity between Sierpiński graphs $S_k^n$ and generalized Sierpiński graphs $S_H^n$ that we use is their fractal structure, see Lemma 7.

First we settle the problem of an extreme vertex. For this we turn to proper 2-extreme vertices, which we will find among vertices of minimum degree. Many of them may not be proper 2-extreme vertices, but an additional condition narrows the choice among them. A vertex $x$ is *controlled* by a vertex $y$ if $N(x) \subset N(y)$, and $x$ is a *twin* of $y$ if $N(x) = N(y)$.

**Lemma 9.** *Let $u$ be a vertex of minimum degree in $S_H^n$, where $n \geq 2$. Then the sum $\sum_{v \in N(u)} \delta(v)$ is maximum among all vertices of minimum degree of $S_H^n$ if and only if one of the following conditions is satisfied:*

*(i) $u$ is a proper 2-extreme vertex,*

*(ii) $u$ is a twin of a proper 2-extreme vertex,*

*(iii) $u$ is controlled by a 2-extreme vertex.*

*Proof.* Let $u$ be a vertex of minimum degree in $S_H^n$, $n \geq 2$. If $u = u_n \dots u_1$ is not a proper 2-extreme vertex, not a twin of a proper 2-extreme vertex and not controlled by any 2-extreme vertex, then at least one of the neighbors of $u$, say $x$, has the same degree in $H$ and in $S_H^n$. On the other hand, an extreme vertex $u_1^n$ is a vertex of minimum degree in $S_H^n$ and the degree of its neighbors in $S_H^n$ is

larger by 1 compared to the degree in $H$. Hence $\sum_{v\in N(u)}\delta(v) < \sum_{v\in N(u_1^n)}\delta(v)$ and $\sum_{v\in N(u)}\delta(v)$ is not maximum among all vertices of minimum degree of $S_H^n$.

Conversely, suppose that the sum $\sum_{v\in N(u)}\delta(v)$ is not maximum for a vertex $u$ of minimum degree in $S_H^n$. This implies that there exists at least one neighbor, say $y$, of $u$ of the same degree in both $H$ and $S_H^n$. In other words, $u$ is not a proper 2-extreme vertex and $u$ is not a twin of a 2-extreme vertex and $u$ is not controlled by a 2-extreme vertex, as the degree of all neighbors of such vertices in $S_H^n$ is lager by 1 compared to the degree in $H$. □

The next lemma is crucial to obtain a base graph from $S_H^n$, where $n > 1$. We will denote by $u'$ the vertex from $H$ which corresponds to a vertex $u \in V(S_H^n)$. By $e_H(u')$ we denote the eccentricity of a vertex $u'$ in a graph $H$. Recall that *eccentricity* of a vertex $u'$ is $e_H(u') = \max\{d_H(u', v') : v' \in V(H)\}$.

**Lemma 10.** *We can obtain a base graph $H$ from $S_H^n$, $n \geq 2$, and a proper 2-extreme vertex $u$ by checking $e_H(u')$ BFS levels starting from $u$.*

*Proof.* Let $u$ be a proper 2-extreme vertex of $S_H^n$ and let us assume without loss of generality that $u = \underline{s}11$, $\underline{s} \in [n_H]^{n-2}$. Let $v_1, \ldots, v_k$, $k = \delta_H(1)$, be the neighbors of 1 in $H$. Clearly all edges between $\underline{s}1v_i$ and $\underline{s}v_i1$, $1 \leq i \leq k$, separate $H_u$ from the rest of $S_H^n$. Moreover, $\underline{s}v_iS_H^1$ is isomorphic to $H$ and there exists an isomorphism $\phi_i : H_u \to \underline{s}v_iS_H^1$ that maps $u$ into $\underline{s}v_i1$ for every $1 \leq i \leq k$. Notice that $H_u$ is contained in the distance levels up to level $\ell_{e(u')}(u)$, but not $\underline{s}v_iS_H^1$, because vertex $\underline{s}v_i1$ is in $\ell_2$ and $\phi_i(u) = \underline{s}v_i1$.

To distinguish which vertices belong to $H \cong H_u$ and which do not, we add an additional information to the BFS algorithm. To each vertex after level $\ell_2$ we keep a pointer to his $\ell_2$-ancestor. All vertices with such an ancestor of the form $\underline{s}v_i1$ are not in $H_u$, but all the others are. □

If an arbitrary graph $G$ is given, then the vertices of $G$ are not labeled as the vertices of $S_H^n$ and we cannot determine whether the corresponding $\ell_2$-ancestor is of the form $\underline{s}v_i1$ or not, as in the proof of Lemma 10. The next lemma settles this problem. If the distance level $\ell_i$, $i > 2$, contains a vertex $v$ with an $\ell_2$-ancestor $x = a_2(v)$, then we say that $x$ is an *active $\ell_2$-ancestor* (in level $\ell_i$), otherwise $x$ is a *non-active $\ell_2$-ancestor* (in level $\ell_i$).

**Lemma 11.** *Let $u$ be a proper 2-extreme vertex of $S_H^n$. If we start our algorithm in $u$ with the additional information about $a_2(v)$, then we have exactly $k = \delta_H(u)$ different active $\ell_2$-ancestors in $\ell_{e_H(u')+1}(u)$ of degree $\delta_H(u) + 1$.*

*Proof.* Without loss of generality we may assume that $u = \underline{s}11$ and that $v_1, \ldots v_k$, $k = \delta_H(1)$, are the neighbors of 1 in $H$. By Lemma 10 all vertices of $H_u$ have been observed by our algorithm before level $\ell_{e_H(u')+1}(u)$. Hence, $\ell_{e_H(u')+1}(u)$ contains no vertices of $H_u$. Since edges between $\underline{s}1v_i$ and $\underline{s}v_i1$, $1 \leq i \leq k$, separate $H_u$ from the rest of $S_H^n$, there will be no $\ell_2$-ancestors from $H_u$ which are active in $\ell_{e_H(u')+1}(u)$. Therefore all active $\ell_2$-ancestors in $\ell_{e_H(u')+1}(u)$ are of the form $\underline{s}v_i1$ and we have at most $k$ active $\ell_2$-ancestors.

To see that there are exactly $k$ active $\ell_2$-ancestors in $\ell_{e_H(u')+1}(u)$ let $e_H(u) = d_H(u,x)$ for $x \in V(H)$ and let $\phi_i$ be an isomorphism between $H_u$ and $\underline{s}v_iS^1_H$ where $\phi_i(u) = \underline{s}v_i1$. We claim that $\phi_i(x) \in \ell_{e(u')+2}(u)$ and that the $\ell_2$-ancestor of $\phi_i(x)$ is $\underline{s}v_i1$ for every $i \in [k]$. Clearly there exists a BFS order in which this is fulfilled. If this is not true in every BFS order, then there exist $i$ and $j$ such that $1 \leq i < j \leq k$ and that $\phi_j(x)$ has an $\ell_2$-ancestor $\underline{s}v_i1$. In other words, there exists a path from $\underline{s}v_i1$ to $\phi_j(x)$ of distance $d_H(v,x)$. This is not possible as $\phi_i(u) = \underline{s}v_i1$ and every path from $\underline{s}v_iS^1_H$ to $\underline{s}v_jS^1_H$ contains a non-base edge. Hence, we have exactly $k$ different active $\ell_2$-ancestors in $\ell_{e(u')+2}(u)$, and therefore also in $\ell_{e(u')+1}(u)$. The degree is clear as there exists an isomorphism $\phi_y$ between $H_u$ and $H_y$ such that $\phi_y(u) = y$ for every active $\ell_2$-ancestor $y$ in $\ell_{e(u')+1}(u)$. □

By Lemmas 10 and 11 we can find a base graph, provided that a proper 2-extremal vertex is known. To find a proper 2-extremal vertex we can use Lemma 9.

**Algorithm** 3
**Input:** A connected graph $G$.
**Output:** There are two possible outputs:
(a) NO if $G$ is recognized as a graph that is not a generalized Sierpiński graph.
(b) A graph $H$ that is a candidate for a base graph of a generalized Sierpiński graph (together with $\ell_2$-ancestors of a proper 2-extreme vertex).

**Begin**

1. **for** every vertex $u$ of minimum degree with maximum $\sum_{v \in N(u)} \delta(v)$, **do** $S \leftarrow u$;
2. **until** $S \neq \emptyset$ **or** $u$ is a proper 2-extreme vertex **do**
   2.1. $S \leftarrow S - \{u\}$;
   2.2. **until** there are more than $\delta(u)$ active $\ell_2$-ancestors **or** more than $\sqrt{|V(G)|}$ vertices have been observed, **do** a BFS-algorithm from $u$;
   2.3. **if** $u$ has exactly $\delta_H(u)$ active $\ell_2$-ancestors in some level $\ell_k$, **then** $u$ is a proper 2-extreme vertex **and** go to 4;
3. $G$ is not a generalized Sierpiński graph, **output** NO **and** STOP;
4. **if** a vertex from $N_G(u)$ has two or more active down neighbors **or** a vertex with a non-active $\ell_2$-ancestor in $\ell_k$ has a neighbor with active $\ell_2$-ancestor in $\ell_k(u)$, **then** $G$ is not a generalized Sierpiński graph, **output** NO **and** STOP.
5. $H$ is induced by $u$, $N(u)$ and vertices with non-active $\ell_2$-ancestors of $u$, **output** $H$ **and** STOP.

**End**

As suggested by Step 3 of the above algorithm, we can detect many graphs which are not generalized Sierpiński graphs. However, if a graph locally behaves like a generalized Sierpiński graph, we can obtain a candidate for $H$ with this algorithm, but the graph can still be far from a generalized Sierpiński graph.

**Theorem 12.** *If $G \cong S_H^n$, then Algorithm 3 correctly recognize the base graph $H$ and works in time complexity $O((|V(H)| - \delta(H))^n \cdot |E(G)|)$ which is better than $O(|V(G)| \cdot |E(G)|)$.*

*Proof.* Let $G \cong S_H^n$. Correctness of the Algorithm 3 follows from Lemmas 9 and 11. For this observe that if a vertex $u$ of minimum degree with maximum $\sum_{v \in N(u)} \delta(v)$ is controlled by some 2-extreme vertex, then it contains more than $\delta_H(u)$ active $\ell_2$-ancestors in $\ell_k$. The same holds if $u$ is a twin of some improper 2-extreme vertex of $H_u$. Hence, if there are exactly $\delta_H(u)$ active $\ell_2$-ancestors at some step of the algorithm, then $u$ is either a proper 2-extreme vertex or a twin of a such vertex. In addition, every edge between $H_u$ and $G - V(H_u)$ is an up edge for an active $\ell_2$-ancestor and Step 4 of Algorithm 3 exclude all graphs for which this is not the case. Lemma 11 then completes the correctness proof of Algorithm 3.

It is clear that we can initiate the algorithm (check for the degrees) within the prescribed time complexity. For Step 1 notice that even if $H$ is a regular graph, then we have in each copy of $H_u$ in $G$ at most $|V(H)| - \delta(H)$ vertices of minimum degree. Altogether there are at most $(|V(H)| - \delta(H))^n$ vertices of minimum degree in $G$. For each such vertex we use $\delta(G)$ operations, which is within the prescribed time limit. In the worst case for Step 2, the algorithm can go through the entire $G$, which costs $O(|E(G)|)$ time. On the other hand this can be done for at most $(|V(H)| - \delta(H))^n$ vertices of $G$ and we arrive at $O((|V(H)| - \delta(H))^n \cdot |E(G)|)$ time. In Step 4 we are already locally limited to vertices of $H$ as we scan only vertices with non-active $\ell_2$-ancestor in $\ell_k$, and for each such vertex we consider all neighbors, which adds $O(\delta(v))$ operations. We need at most $O(|V(H)| \cdot \Delta(H))$ time for Step 4. For Step 5 we need to check only for the $\ell_2$-ancestors of vertices scanned by a BFS algorithm from $u$. If one forms the set of its descendants for each $\ell_2$-ancestor, then this can clearly be done even faster than claimed. Hence the time complexity of Algorithm 3 is $O((|V(H)| - \delta(H))^n \cdot |E(G)|)$. □

By combining Algorithms 2 and 3, we can recognize any generalized Sierpinski graph $S_H^2$. Its time complexity is polynomial only if the base graph belongs to $\mathcal{IP}$. However, the same approach is not possible for $n > 2$. The reason for this is that every 2-extreme vertex of $S_k^2$ is also an extreme vertex and is not incident to a non-base edge, while for $n > 2$ this does not hold anymore and we are not able to separate between base and non-base edges incident with a 2-extreme vertices.

Another way to recognize an arbitrary generalized Sierpiński graph is also connected with isomorphism checking. We can use Algorithm 3 to obtain a candidate for a base graph $H$ from an arbitrary graph $G$. Then we can compute $n = \log_{|V(H)|} |V(G)|$ (as in Algorithm 2) and construct $S_H^n$. At the end we need to check if $G$ is isomorphic to $S_H^n$. However, here the isomorphism check is done on $G$ and not on smaller $H$.

**Ackwnoledgments.** The authors are grateful to the referees for their valuable comments. First author was partially supported by OEAD Projekt SI 08/2016 and the second author by the grant BI-AT/16-17-024.

## REFERENCES

1. H. Bodlaender: *Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees.* J. Algorithms **11** (1990), 631–643.
2. K. S. Booth, G. S. Lueker: *A linear time algorithm for deciding interval graph isomorphism.* J. of the ACM **26** (1979), 183–195.
3. E. Estaji, J. A. Rodríguez-Velázquez: *The strong metric dimension of generalized Sierpiński graphs with pendant vertices.* Ars Math. Contemp. **12** (2017), 127–134.
4. A. Estrada-Moreno, J. A. Rodríguez-Velázquez, E. D. Rodríguez-Bazan: *On generalized Sierpiński graphs.* Discuss. Math. Graph Theory **37** (2017), 547–560.
5. A. Estrada-Moreno, J. A. Rodríguez-Velázquez: *On the general Randić index of polymeric networks modelled by generalized Sierpiński graphs.* Discrete Appl. Math. **263** (2019), 140–151.
6. A. Estrada-Moreno, E. D. Rodríguez-Bazan, J. A. Rodríguez-Velázquez: *On distances in generalized Sierpiński graphs.* Appl. Anal. Discrete Math. **12** (2018), 49–69.
7. J. Geetha, K. Somasundaram: *Total coloring of generalized Sierpiński graphs.* Australas. J. Combin. **63** (2015), 58–69.
8. S. Gravier, M. Kovše, A. Parreau: *Generalized Sierpiński graphs.* Posters at EuroComb'11, Budapest,
   http://www.renyi.hu/conferences/ec11/posters/parreau.pdf.
9. A. M. Hinz: *The Tower of Hanoi.* Enseign. Math. **(2) 35** (1989), 289–321.
10. A. M. Hinz, C. Holz auf der Heide: *An efficient algorithm to determine all shortest paths in Sierpiński graphs.* Discrete Appl. Math. **177** (2014), 111–120.
11. A. M. Hinz, S. Klavžar, U. Milutinović, D. Parisse and C. Petr: *Metric properties of the Tower of Hanoi graphs and Stern's diatomic sequence.* European J. Combin. **26** (2005), 693–708.
12. A. M. Hinz, S. Klavžar, U. Milutinović, C. Petr: *The Tower of Hanoi–Myths and Maths.* Birkhäuser/Springer, Basel, 2013.
13. A. M. Hinz, S. Klavžar, S. S. Zemljič: *A survey and classification of Sierpiński type graphs.* Discrete Appl. Math. **217** (2017), 565–600.
14. A. M. Hinz, D. Parisse: *The average eccentricity of Sierpiński graphs.* Graphs Combin. **28** (2012), 671–686.
15. J. Hopcroft, J. Wong: *Linear time algorithm for isomorphism of planar graphs.* Proceedings of the Sixth Annual ACM Symposium on Theory of Computing (1974), 172–184.
16. P. J. Kelly: *A congruence theorem for trees.* Pacific J. Math. **7** (1957), 961–968.

17. S. Klavžar, U. Milutinović: *Graphs $S_k^n$ and a variant of the Tower of Hanoi problem.* Czechoslovak Math. J. **47(122)** (1997), 95–104.

18. S. Klavžar, S. S. Zemljič: *Connectivity and some other properties of generalized Sierpiński graphs.* Appl. Anal. Discrete Math. **12** (2018), 401–412.

19. S. Lipscomb: *Fractals and Universal Spaces in Dimension Theory.* Springer, Berlin, 2009.

20. E. M. Luks: *Isomorphism of graphs of bounded valence can be tested in polynomial time.* J. Comput. Syst. Sci. **25** (1982), 42–65.

21. M. Muzychuk: *A Solution of the Isomorphism Problem for Circulant Graphs.* Proc. London Math. Soc. **88** (2004), 1–41.

22. D. Parisse: *On some metric properties of the Sierpiński graphs $S_k^n$.* Ars Combin. **90** (2009), 145–160.

23. F. Ramezani, E. D. Rodríguez-Bazan, J. A. Rodríguez-Velázquez: *On the Roman domination number of generalized Sierpiński graphs.* FILOMAT **31** (2017), 6515–6528.

24. J. A. Rodríguez-Velázquez, J. Tomás-Andreu: *On the Randić index of polymeric networks modelled by generalized Sierpiński graphs.* MATCH Commun. Math. Comput. Chem. **74** (2015), 145–160.

**Wilfried Imrich**
Montanuniversität Leoben
Franz Josef-Straße 18, 8700 Leoben, Austria
E-mail: *imrich@unileoben.ac.at*

(Received 31.03.2018)
(Revised 22.12.2020)

**Iztok Peterin**
University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Koroška 46, 2000 Maribor, Slovenia
E-mail: *iztok.peterin@um.si*