mathematics

MDPI

Article

Research of NP-Complete Problems in the Class of Prefractal Graphs

Rasul Kochkarov

Department of Data Analysis and Machine Learning, Faculty of Information Technology and Big Data Analysis, Financial University under the Government of the Russian Federation, Leningradsky Prospekt, 49, 125993 Moscow, Russia; rasul_kochkarov@mail.ru

Abstract: NP-complete problems in graphs, such as enumeration and the selection of subgraphs with given characteristics, become especially relevant for large graphs and networks. Herein, particular statements with constraints are proposed to solve such problems, and subclasses of graphs are distinguished. We propose a class of prefractal graphs and review particular statements of NP-complete problems. As an example, algorithms for searching for spanning trees and packing bipartite graphs are proposed. The developed algorithms are polynomial and based on well-known algorithms and are used in the form of procedures. We propose to use the class of prefractal graphs as a tool for studying NP-complete problems and identifying conditions for their solvability. Using prefractal graphs for the modeling of large graphs and networks, it is possible to obtain approximate solutions, and some exact solutions, for problems on natural objects—social networks, transport networks, etc.

Keywords: NP-complete problems; prefractal graphs; subgraph search algorithms

Citation: Kochkarov, R. Research of NP-Complete Problems in the Class of Prefractal Graphs. *Mathematics* **2021**, *9*, 2764. https://doi.org/10.3390/math9212764

Academic Editors: Jonatan Lerga, Ismael Gonzalez Yero, Ljubisa Stankovic, Nicoletta Saulig and Cornel Ioana

Received: 23 September 2021
Accepted: 29 October 2021
Published: 31 October 2021

1. Introduction

The beginning of the study of intractable problems is associated with the possibility of eliminating enumeration to find the optimal solution and create a polynomial algorithm. In the case of an exponential number of variants of solutions, the exhaustive algorithm does not allow one to find the optimal solution in an acceptable time and becomes intractable or even unsolvable. The efficiency of finding a solution and the complexity of the algorithm are estimated based on the solution time, limited by a function of the size of the problem. In discrete mathematics, two classes of problems are considered: the NP class of all exhaustive problems and the class P of exhaustive problems solvable in polynomial time (in classical terminology, with a Turing machine). In the NP class, typical (NP-complete) problems that set the "standard" of complexity are distinguished. Any problem from NP reduces polynomially to an NP-complete problem [1].

In [2], using the examples of the simple max cut, node cover, and directed Hamiltonian path problems, it was shown that a number of NP-complete problems remain NP-complete even when their domains are substantially restricted. In [3], a special class of indefinite quadratic programs with simple constraints and integer data was proposed. It was shown that the problem of checking that a given feasible solution is not a local minimum (or that the objective function is not bounded from below on the set of feasible solutions) in this class is NP-complete. In [4] some new graph problems (the optimal linear cut arrangement, optimal directed tree arrangement and arrangements on a grid) were added to the list of known NP-complete problems. Despite the significant amount of work on NP-completeness in the past years, research on this topic remains relevant. This also applies to graph problems. Among the modern approaches are the following. To study such NP-complete problems as the min dominating set, max independent set, max clique, etc., the effectiveness and ineffectiveness of all nodes in the given graph are computed in [5]. The use of modern methods and capabilities of computer science for solving well-known

graph NP-complete problems should be noted. For example, a new iteration-parallel-based method was been used for reconfigurable computer systems [6]. Graph-theoretic intractable problems are also common in application areas. For example, the study of a hierarchical subsystem decomposition problem with a genetic algorithm allows for a better understanding of large-scale software systems [7].

For solving NP-complete problems, it is customary to define subclasses of graphs for which solvability conditions are possible. Another approach is the formulation of particular problems with additional restrictions [8–10]. NP-complete problems are explored in various subject areas to find solutions to applied problems [11–13].

The prospects and significance of research related to fractal sets can be assessed by means of regular conferences and periodicals. For example, the magazine *Chaos, Solitons & Fractals* is entirely devoted to this topic [14–16]. This allows us to speak about the range of model problems based on fractal sets. Among these, problems and models have been distinguished, in which fractal sets are presented as self-similar (fractal) graphs of large dimensions. In addition, fractal graphs often act as models of structures of complex multielement systems such as communication networks. Research is carried out in three main areas: the recognition of fractal graphs, the properties and characteristics of fractal graphs, and multicriteria optimization problems.

Self-similar graphs are a subclass of prefractal graphs. Various authors have introduced independent definitions of self-similar graphs and have called them families. They have considered special cases of families of self-similar graphs, such as Farey graphs, 2-dimensional Sierpiński gasket graphs, Hanoi graphs, modified Koch graphs, Apollonian graphs, pseudofractal scale-free webs, fractal scale-free networks, etc. [17–19]. In our terminology, the graphs of these families are noncanonical prefractal graphs, for which special generation conditions are specified.

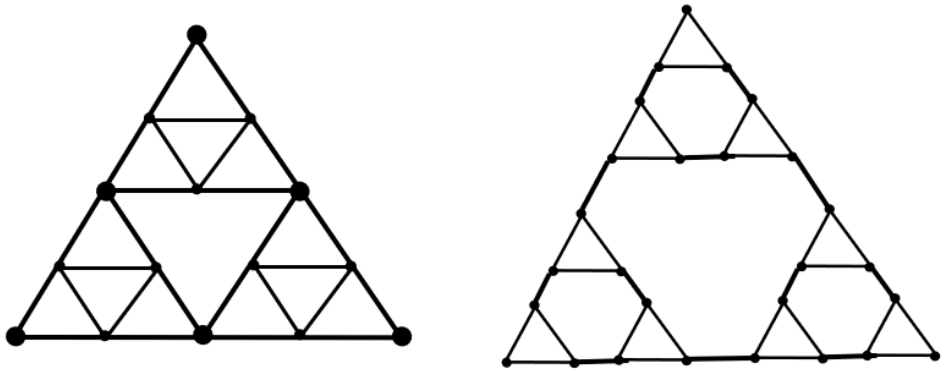
Prefractal graphs [20–22] represent a relatively new subclass of large dynamic graphs [23–25]. With large prefractal graphs, it is possible to build graph-theoretic models of the structure of social networks and solve various optimization problems on them [26–28]—finding the shortest paths, highlighting subgraphs, multicriteria optimization, etc. Separate tasks include the visualization of a dynamic graph and the generation of a sequence of a dynamic graph with the preservation of characteristics and properties, including the stability of solutions when moving through the sequence [29,30].

The concepts of fractal and prefractal graphs should be considered separately. Since fractal graphs are infinite, more research is required, the result of which should be methods of visual display, the storage of information about the structure of a fractal graphs, the possibility of compressing this information, etc. At the same time, a fractal graph is a continuation of a prefractal graph. On the other hand, a sequence of prefractal graphs is a dynamic graph. This direction also requires additional study.

In this paper, attention is paid to the class of prefractal graphs, as graphs with self-similar properties. A flexible procedure for generating prefractal graphs allows one to combine in this class families of self-similar graphs, for which different authors introduce separate generation rules. As mentioned above, such families belong to noncanonical prefractal graphs. Figure 1 shows noncanonical G'_3 (a) and canonical G_3 (b) prefractal graphs. A noncanonical prefractal graph (a) corresponds to the Sierpinski triangle in the terminology of self-similar graphs.

It should be noted that the first studies of Sierpinski carpets were carried out back in the 1990s [31,32]. In those publications, geometric objects were considered as one of the types of fractals [33]. Later, in the 2000s, publications appeared offering definitions of Sierpinski graphs from the classical point of view, when a graph is given sets of vertices and edges [34–36]. Recently, researchers have proposed a uniform definition of Sierpinski graphs [37,38]. As mentioned above, Sierpinski graphs belong to the class of prefractal graphs, and for which specific generation rules are given, that is, Sierpinski graphs are also prefractal graphs. All results obtained for Sierpinski graphs apply to the corresponding

type of prefractal graphs, including optimization problems—finding the shortest paths, Hamiltonian subgraphs, coloring, etc. [39–41].



(**a**) G_3' (**b**) G_3

Figure 1. (**a**) Noncanonical prefractal graph G_3'; (**b**) canonical prefractal graph G_3.

Sierpinski graphs generated by a triangle (complete 3-vertex graph) or square (complete 4-vertex graph or 4-vertex cycle) are often considered. Following the rules for generating a prefractal graph, both simple graphs with 3–4 vertices and complex graphs with a large number of vertices and connections can act as a basis, where it is possible to specify the preservation of the edge adjacency or to specify an arbitrary adjacency. In the case of weighing a prefractal graph, a similarity coefficient is set, which changes the weights of the edges at each step. The following are some commonly used definitions of prefractal graphs.

The generally accepted definitions and notation $G = (V, E)$ for graphs are used [42,43]. Most of the definitions of a class of prefractal graphs are shown in [44,45]. A seed is a connected graph $H = (W, Q)$ with unlabeled vertices $v \in W$. Seeds—also called graphlets—are small connected non-isomorphic induced subgraphs of a large graph or network [46–48]. Graphlets were first used to design highly sensitive measures of network local structural similarities [49]. However, graphlets are used more often in specific formulations of problems; therefore, in this work, a general definition of a seed is used.

A prefractal graph is denoted as $G_L = (V_L, E_L)$, where V_L is the set of vertices, and E_L is the set of edges. In what follows, a simplified notation G_L is used for known (canonical) prefractal graphs $G_L = (V_L, E_L)$. In the process of constructing a prefractal graph, a trajectory is formed $G_1, G_2, \ldots, G_L$. The graph constructed at step $l = 1, 2, \ldots, L$ is called a prefractal graph G_l of rank l. The new edges of the graph G_L are the edges of rank L, and the remaining edges are the old edges of the rank l.

As $l \to \infty$, the graph G_l is fractal. The first mentions of fractal graphs can be found in [50]. A fractal graph, like a fractal, is an infinite object. For a fixed value of rank l, a pre-fractal graph is considered. For example, as shown above for $l = L$ the graph G_L is considered.

Figure 2 shows an example of replacing the vertices of a graph G_1 with a full 3-vertex seed H with an arbitrary adjacency of old edges: (a) small dashed circles outline the vertices replaced by the seed; (b) the middle dashed circles outline the seeds that replace the vertices; (c) old edges of the graph are marked with bold lines.

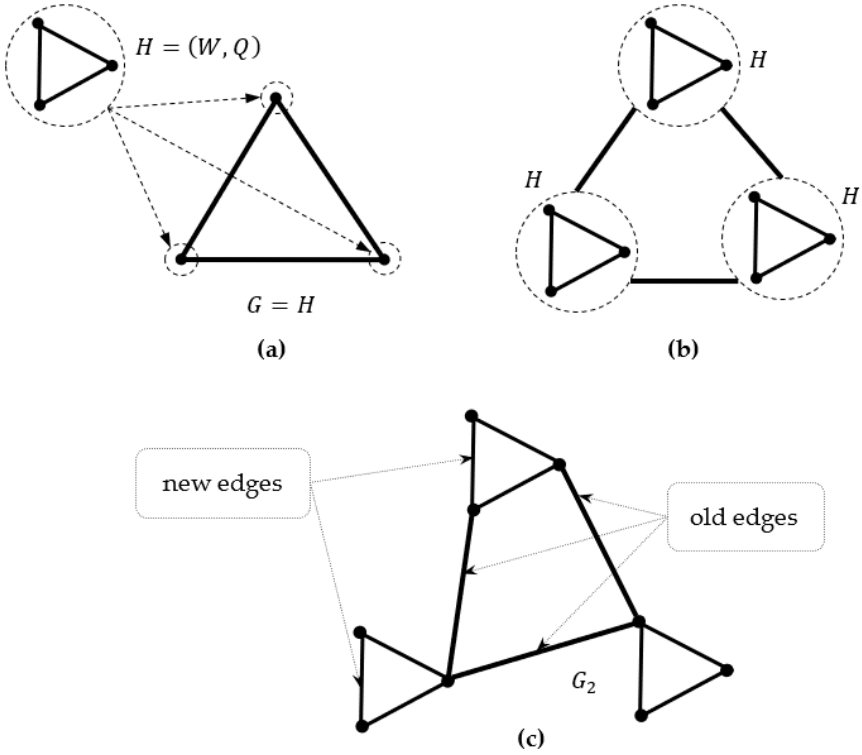


Figure 2. Operation of the replacement of vertices of graph G_1 by seed H.

The essential characteristics of a prefractal graph $G_L = (V_L, E_L)$ are the number of its vertices (1) and edges (2).

$$N = N(n, L) = |V_L| = n^L, \tag{1}$$

where $n = |W|$ is the number of vertices of seed H.

$$M = M(n, q, L) = |E_L| = q\left(1 + n + n^2 + \cdots + n^{L-1}\right) = q\left(n^L - 1\right)/(n-1), \tag{2}$$

where $q = |Q|$ is the number of edges of seed H.

This article explores some NP-complete problems in the class of prefractal graphs and conditions for their solvability for particular statements. The purpose of this work is to lay the foundations for creating rules for determining conditions for the polynomial solvability of problems on prefractal graphs, and also to acquaint readers with these kinds of graphs as prefractal graphs.

Further, we present new theorems for particular problems in the class of prefractal graphs. The classical formulations of intractable problems are taken from [1].

2. Particular Problems in the Class of Prefractal Graphs

2.1. Subgraph Isomorphism

INSTANCE: Graphs $G = (V_1, E_1)$, $H = (V_2, E_2)$.

QUESTION: Does G contain a subgraph isomorphic to H?

An algorithm for the minimal numbering of the vertices is proposed to formulate and prove the theorem on the question of isomorphism. Let prefractal graph G_L be generated by a complete seed, preserving the adjacency of the old edges.

Figure 3 shows a prefractal graph G_3 generated by a 4-vertex seed—a complete graph where old edges are adjacent. The bold lines mark the edges of the seed of the first rank, and the dashed circles outline the seed of the second and third ranks. Following Algorithm 1 (NUM), the vertices of the prefractal graph G_3 are numbered by $n = 4$.

Algorithm 1 Algorithm for the Minimal Numbering of the Vertices (NUM)

Input : prefractal graph $G_L = (V_L, E_L)$.
Output : the minimum numbering of the vertices of G_L by a number n.

1. Sequentially number the vertices of the seed $z_1^{(1)}$ of the first rank with numbers $1, 2, \ldots, n$.
2. At the output of step 1, one vertex of the seed $z_s^{(2)}$ of the second rank is numbered, which also belongs to the seed $z_1^{(1)}$.

Sequentially number the remaining $(n-1)$ vertices of $z_s^{(2)}$, $s = \overline{1,n}$ with numbers from the set $\{1, 2, \ldots, n\}$ different from the already numbered vertices.
for $l = 3$ to L do:
l. At the output of step $l-1$, one vertex of the seed $z_s^{(l)}$ of the rank l is numbered, which also belongs to the seed $z_s^{(l-1)}$.

Sequentially number the remaining $(n-1)$ vertices of $z_s^{(l)}$, $s = \overline{1,n^{l-1}}$ with numbers from the set $\{1, 2, \ldots, n\}$ different from the already numbered vertices.

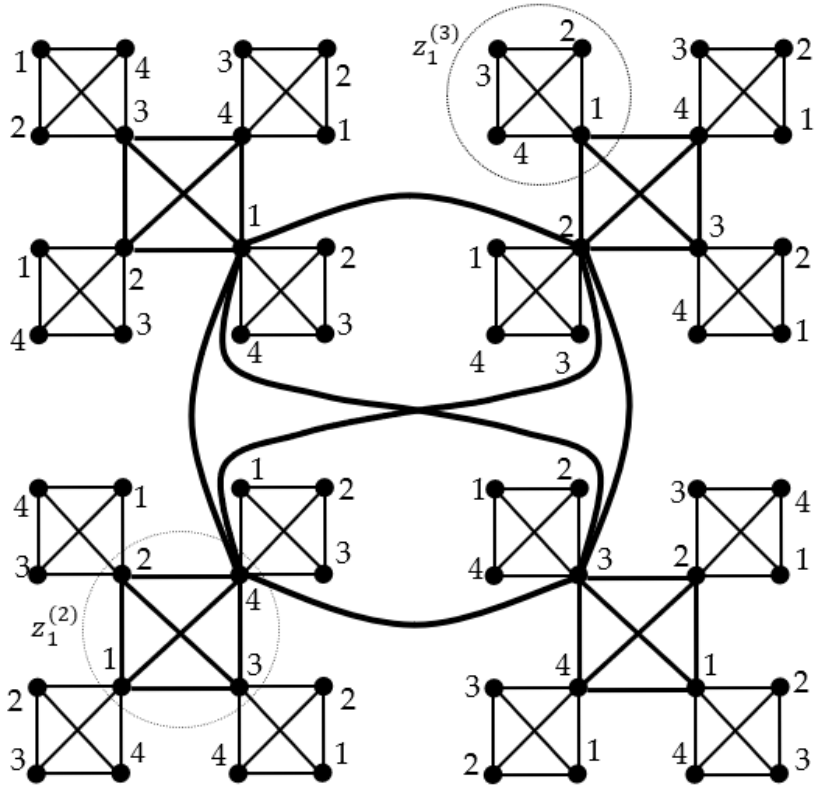


Figure 3. Minimum numbering of vertices of a prefractal graph G_3 generated by a complete 4-vertex seed, where old edges are adjacent.

Theorem 1. *Algorithm 1 enumerates the vertices of the prefractal graph generated by the complete seed with minimal numbers, while preserving the adjacence of the old edges.*

Proof of Theorem 1. Since the seed is a complete graph, the minimum possible number for numbering the vertices is at least $n = |W|$. At the first step, the vertices of the graph G_1 or, in another way, the seed $z_1^{(1)}$ of the first rank are numbered with $1, 2, \ldots, n$. In the second step, we consider a graph G_2 in which the n vertices are already numbered. Each vertex of $z_1^{(1)}$ is included in only one seed of the second rank $z_s^{(2)}$, $s = \overline{1,n}$ due to the preserving of the adjacency of the old edges of the first rank. The vertices common for $z_1^{(1)}$ and $z_s^{(2)}$ are already numbered, that is, in each seed $z_s^{(2)}$ one vertex is numbered with a number from $\{1, 2, \ldots, n\}$. Then the remaining $n-1$ vertices of $z_s^{(2)}$ are numbered differently from those

already numbered in $z_1^{(1)}$. Since the seed is a complete graph, the vertices of $z_s^{(2)}$ can be numbered at least with n. At the following steps $l = 3, 4, \ldots, L$ the order of numbering is repeated, in each G_l seeds $z_s^{(l)}$, $s = \overline{1, n^{l-1}}$ are added. Since the added seed $z_s^{(l)}$ is a complete graph, the minimum possible number for numbering the vertices of G_l is at least n. Furthermore, the preservation of the adjacence of the old edges of G_l guarantees the numbering of the vertices, the newly added seeds, by numbers $1 + (n-1) = n$. □

Consequence 1. *Algorithm 1 enumerates the vertices of a prefractal graph, the adjacency of the old edges of which is preserved, no more than n numbers.*

Theorem 2. *Any two prefractal graphs $G_L^1 = (V_L^1, E_L^1)$ and $G_L^2 = (V_L^2, E_L^2)$ are isomorphic if the adjacence of the old edges is preserved, the seeds $H_1 = (W_1, Q_1)$, $H_2 = (W_2, Q_2)$ that generate them are complete graphs, and $|W_1| = |W_2|$.*

Proof of Theorem 2. Following the definition of isomorphism of two graphs, it is necessary to set a one-to-one mapping $f : V_L^1 \to V_L^2$ such that any two vertices u and v of G_L^1 are adjacent if and only if the vertices $f(u)$ and $f(v)$ are adjacent in G_L^2. In other words, if there is an edge $e \in E_L^1$ between two vertices in G_L^1, then there is an edge $f(e) \in E_L^2$ in G_L^2.

Seeds H_1, H_2 and, accordingly, graphs G_1^1, G_1^2 are isomorphic since they are complete graphs according to the conditions of the theorem. Using Algorithm 1, the vertices of G_1^1 and G_1^2 are numbered with the minimum number $n = |W_1| = |W_2|$.

Next, we consider the graphs G_2^1, G_2^2 of the second rank in the trajectory $l = 1, 2, \ldots, L$. Seeds of the first rank are isomorphic due to the isomorphism G_1^1 and G_1^2. Seeds of the second rank are isomorphic due to the isomorphism H_1 and H_2. Thus, for any edge between a pair of vertices in G_2^1, there is an edge between a pair of vertices in G_2^2 that is confirmed by the numbering of the vertices of both graphs.

The adjacency of the old edges in G_L^1, G_L^2 and the generation by complete n-vertex seeds in the trajectory $l = 1, 2, \ldots, L$ provides, due to isomorphism H_1 and H_2, the isomorphism of prefractal graphs. In other words, at each step of the trajectory $l = 1, 2, \ldots, L$, isomorphic seeds are added to the isomorphic graphs G_l^1, G_l^2. □

Consequence 2. *Any two prefractal graphs G_l^1, G_l^2 are isomorphic in the trajectory $l = 1, 2, \ldots, L$, if the adjacency of the old edges is preserved, and the seeds that generate them are complete n-vertex graphs.*

Theorem 2 deals with unweighted prefractal graphs. In the case of prefractal graphs weighted by the similarity coefficient, we will talk about interval isomorphism when the weights of two compared edges lie on the same interval.

A prefractal graph $G_L = (V_L, E_L)$ is called weighted if a real number $w(e^l) \in \left(\theta^{l-1}a, \theta^{l-1}b\right)$ is assigned to each of its edges $e^l \in E_L$, where $l = 1, 2, \ldots, L$ is the rank of the edge, $a > 0$, $\theta < a/b$.

Consequence 3. *Any two weighted prefractal graphs are interval isomorphic if the adjacency of the old edges is preserved; the seeds that generate them are complete n-vertex graphs.*

2.2. Degree Constrained Spanning Tree

INSTANCE: Graph $G = (V, E)$, positive integer $K \leq |V|$.

QUESTION: Is there a spanning tree for G in which no vertex has degree larger than K?

An algorithm for finding a spanning tree is proposed to formulate and prove the theorem on the degree constrained spanning tree. Let prefractal graph G_L be generated with an arbitrary adjacency of the old edges.

As a procedure, instead of Prim's algorithm, it is possible to use any other known algorithms for finding a spanning tree on a graph.

Figure 4 shows a prefractal graph G_3 generated by the 4-vertex seed—a complete graph, the old edges of which are not adjacent. At the first step of the algorithm

(Figure 4a) spanning trees of the seeds of the third rank are selected, at the second step (Figure 4b) the spanning trees of the second rank seeds are selected. At the third step (Figure 4c), the spanning tree is selected on the graph G_1, which corresponds to the selection of the spanning tree of the entire prefractal graph G_3.

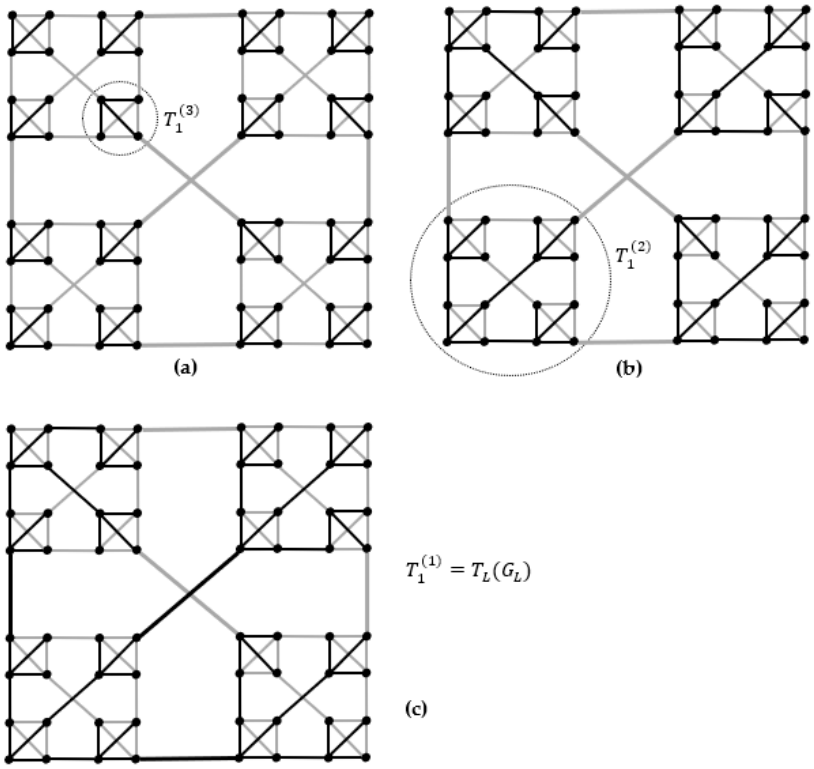


Figure 4. The spanning tree of the entire prefractal graph G_3 generated by a complete 4-vertex seed, the old edges of which are not adjacent.

Theorem 3. *Algorithm 2 finds the spanning tree of the prefractal graph.*

Algorithm 2 Algorithm for Finding a Spanning Tree (sTREE)

Input: prefractal graph $G_L = (V_L, E_L)$.
Output: spanning tree $T_L = (V_L, E_L^*)$ of G_L.
for $l = L$ to 1 do:
for $s = 1$ to n^{l-1} do:
$L - l + 1.s$. For each seed $z_s^{(l)}$ find a spanning tree $T_s^{(l)}$ using Prim's procedure.
2. At the output of step L, $k = (n^L - 1)/(n - 1)$ spanning trees $T_s^{(l)}$ for each seed $z_s^{(l)}$ are selected. Selecting all spanning trees corresponds to the spanning tree T_L of the prefractal graph G_L.
Prim's Procedure. Procedure for Finding a Spanning Tree (MST)
Input: graph $G = (V, E)$.
Output: spanning tree $T = (V, E^*)$ of G.

Proof of Theorem 3. Algorithm 2 finds the spanning trees $T_s^{(l)}$, $l = 1, 2, \ldots, L$, $s = \overline{1, n^{l-1}}$ in G_L. Prim's procedure is used in the classical form to find the spanning tree of an arbitrary graph. Spanning trees $T_s^{(L)}$, $s = \overline{1, n^{L-1}}$ selected on the seeds $z_s^{(L)}$ form a spanning forest consisting of connected n^{L-1} components—blocks of the first rank. Next, $T_s^{(L-1)}$, $s = \overline{1, n^{L-2}}$ selected on $z_s^{(L-1)}$ form a spanning forest of connected n^{L-2} components—blocks of the second rank. Since spanning trees are added to spanning trees from high to low rank through one vertex, the new components are also spanning trees without cycles. Thus, passing step-by-step from L to the second rank, we obtain n spanning trees—blocks

of the $(L-1)$ rank. At the last step, the spanning tree $T_1^{(1)}$ of the seed $z_1^{(1)}$ connects n components into one spanning tree T_L of G_L. □

Remark 1. *Algorithm 2 selects the spanning trees in the trajectory* $G_1, G_2, \ldots, G_L$.

Figure 5 shows the steps of the sequential selection of spanning trees for G_1, G_2, G_3. At the first step, a spanning tree is selected for G_1 and the same edges are highlighted by a bold line on G_3 (a). At the second and third steps, the spanning trees for G_2 (b), G_3 (c) are respectively selected.

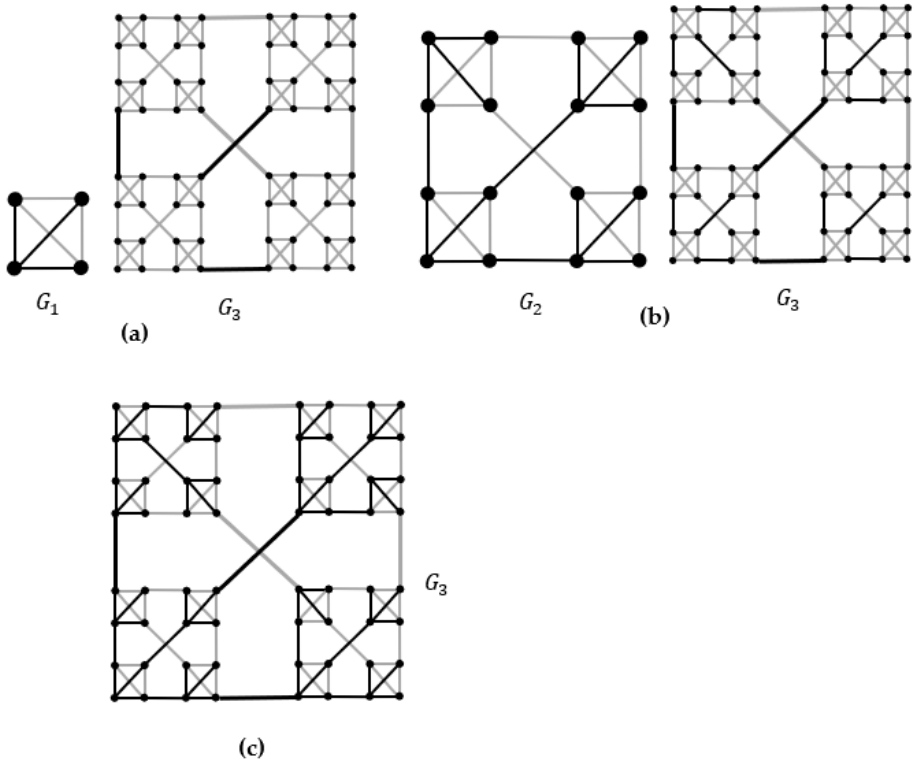


Figure 5. The of spanning trees for G_1, G_2 and G_3.

Consequence 4. *Algorithm 2 selects a minimum weight spanning tree of a prefractal graph weighted by a similarity coefficient* $\theta < a/b$.

Prim's procedure uses the weights of the spanning trees obtained in the previous steps and maintains a minimum weight at each rank $l = L, L-1, \ldots, 1$. At rank L, the procedure corresponds to the classical Prim algorithm. At rank $L-1$, the procedure uses the spanning trees obtained in the seeds of rank L. Each vertex in the seeds of the rank $L-1$ is assigned a weight of the minimum spanning tree of the corresponding seed of the rank L. Furthermore, the search for the minimum spanning tree of the current seed of the rank $L-1$ is carried out, taking into account the total weight of the vertex-edge-vertex.

Figure 6 shows an example of the calculation of the total weight of a vertex-edge-vertex. To search for the minimum spanning tree $z_1^{(2)}$ the weights of the vertices v', v'', v''' are used, $w(v') = w(T_1^{(3)})$, $w(v'') = w(T_2^{(3)})$, $w(v''') = w(T_3^{(3)})$. Prim's procedure uses the total weight of an edge e'': $w(v') + w(e'') + w(v'')$. It is assumed that the prefractal graph is connected; otherwise, the algorithm will result in a spanning forest of minimum weight.

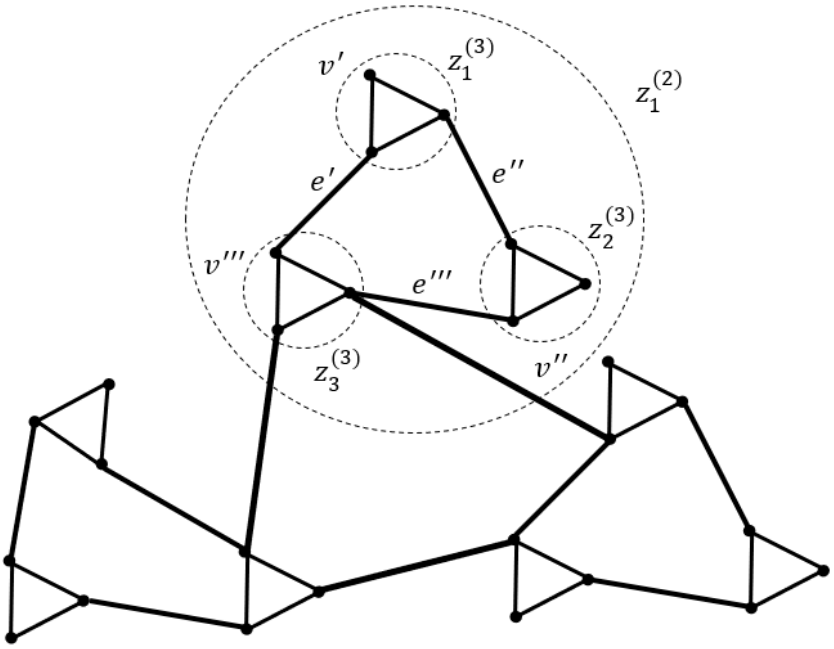


Figure 6. Prefractal graph G_3 the old edges of which are not adjacent.

Consequence 5. *Algorithm 2 takes $O(c{\cdot}N)$ time, where $N = n^l$ and parameter $c = 4n^2$.*

At each step, the algorithm selects the minimum spanning trees for $z_s^{(l)}$, $s = \overline{1, n^{L-1}}$. Prim's algorithm takes less than $O(n^2)$ time. Prim's procedure includes two additional operations for adding the weights of the vertices: $(2n)^2 = 4n^2$. In total, one will perform $(n^L - 1)/(n-1){\cdot}2n^2 \leq n^L{\cdot}4n^2 = 4n^2{\cdot}N$ operations.

Theorem 4. *If the old edges are not adjacent, there is a spanning tree for any prefractal graph G_L in which no vertex has a degree larger than $K = k + 1$, where k—the maximum degree of vertices of a spanning tree of the seed.*

Proof of Theorem 4. The maximum degree of vertices of a spanning tree of seed H is k following the conditions of the theorem. In the process of generating a prefractal graph at each step $l = 1, 2, \ldots, L$, each vertex can be incident to only one old edge due to the condition that any old edges are not adjacent. Thus, any vertex of spanning tree in the trajectory $G_1, G_2, \ldots, G_L$ has a degree k or $k + 1$ in the case of incidence with the old edge. In other words, the spanning tree of G_L contains only vertices with degrees at most $K = k + 1$. □

Consequence 6. *If the old edges are not adjacent, there is a spanning tree for any prefractal graph G_l, $l = 1, 2, \ldots, L$ in which no vertex has a degree larger than $K = k + 1$.*

Remark 2. *If the old edges are adjacent, there is a spanning tree for any prefractal graph G_L in which vertexes have a degree from k to k·L.*

2.3. Maximum Leaf Spanning Tree

INSTANCE: Graph $G = (V, E)$, positive integer $K \leq |V|$.
QUESTION: Is there a spanning tree for G in which K or more vertices have degree 1?

Lemma 5. *Leaf vertices of any spanning tree of G_L are placed in seeds of the rank L.*

Proof of Lemma 5. Let a prefractal graph generated by a seed be a connected graph. Otherwise, since the prefractal graph will consist of several connected components, it is impossible to select a connected spanning tree. Following the procedure for replacing

vertices with a seed, each vertex of G_1 is replaced with a seed H. The old edges are adjacent to the new edges in an arbitrary order. In G_2 each vertex is incident with new edges, and some of the vertexes are incident with old edges. Thus, vertices incident to both old edges and new ones cannot be leaf vertices. Only the vertices incident to the new edges are leaf vertices. Continuing the procedure for generating a prefractal graph, at each step $l = 1, 2, \ldots, L$ the vertices are replaced with seed. Vertices of G_{L-1} were replaced by seeds, and only vertices incident to new edges can be a leaf. In other words, the leaf vertices of G_L are placed in seeds of the rank L. In the case of generating a spanning tree, leaf vertices are also placed in seeds of the rank L, as in any arbitrary case described above.

Consider the case of the selection of a spanning tree of an arbitrary G_L. Suppose that any leaf vertex is incident only to the old edge, but this contradicts the definition of a connected prefractal graph. Any vertex is incident to a new edge. Thus, any vertex incident to the old edge is also incident to the new edge and is not a leaf vertex, and the leaf vertex of the spanning tree can only be incident to a new edge. In other words, leaf vertices of any spanning tree of a prefractal graph are placed in seeds of the rank L. □

Theorem 6. *If there is a spanning tree for the seed H with at least $k \leq n$ leaf vertices and the adjacency of old edges is preserved, then there is a spanning tree for the prefractal graph G_l, $l = 1, 2, \ldots, L$ with at least $K = (k-1)\cdot n^{l-1}$ leaf vertices.*

Proof of Theorem 6. According to the lemma, the leaf vertices of any spanning tree are placed in the seeds of the rank L. Let us consider step-by-step the procedure for generating a prefractal graph and selecting its spanning tree.

There is a spanning tree $T_1 = T_1^1$ of $G_1 = H$ with at least $k \leq n$ leaf vertices according to the hypothesis of the theorem. At the next step, all vertices G_1 are replaced with a seed H and G_2 is formed while preserving the adjacence of the old edges. The same goes for T_1. The spanning trees T_s^2 in seeds of the second rank and T_1 forms spanning tree T_2 in G_2. Only leaf vertices in T_s^2 can be leaf vertices in T_2. T_s^2 are actually attached to T_1 by one vertex. In the minimal case, T_s^2 can be attached with its leaf vertex, then the number of leaf vertices of T_2 is equal to $(k-1)\cdot n$. By selecting T_s^l in the trajectory $l = 1, 2, \ldots, L$ the spanning trees T_l of G_l are formed. Thus, a spanning tree T_l with at least $(k-1)\cdot n^{l-1}$ vertices are selected for each G_l, $l = 1, 2, \ldots, L$. □

Consequence 7. *For any G_l, $l = 1, 2, \ldots, L$ where old edges are not adjacent, there are at most $K = n^l - 2n^{l-1} + 2$ leaf vertices.*

When the old edges are not adjacent and the seed is the star, there is the maximum increase of leaf vertices in G_L. Then there are $(n-1)$ leaf vertices in G_1. The old edges are at first not incident to leaf vertices, and then are incident to leaf vertices. Then there are $n^2 - 2(n-1)$ leaf vertices in G_2 and $n\left(n^2 - 2(n-1)\right) - 2(n-1)$ in G_3. Furthermore, $K = n^l - 2n^{l-1} + 2$ leaf vertices remain in G_l, $l = 1, 2, \ldots, L$.

Consequence 8. *When the conditions of Theorem 6 are satisfied, the parameterized polynomial algorithm selects a spanning tree for G_l with at least $K = (k-1)\cdot n^{l-1}$ leaf vertices in time $O(c\cdot N)$, where $N = n^l$ and parameter $c = 2^n$.*

The selection of spanning trees with the maximum number of leaf vertices in seeds can be carried out by means of any known algorithm. In the worst case, it will require 2^n operations. In total, there are $n^l = N$ seeds in G_l.

Figure 7 shows several examples of the generation of spanning trees, and the numbers of leaf vertices for them are calculated:

(a) $K_3 = 2 \leq K = 2^3 - 2\cdot 2^2 + 2 = 2$;
(b) $K_3 = 33 \leq K = 4^3 - 2\cdot 4^2 + 2 = 34$;
(c) $K_3 = 132 \leq K = 6^3 - 2\cdot 6^2 + 2 = 146$.

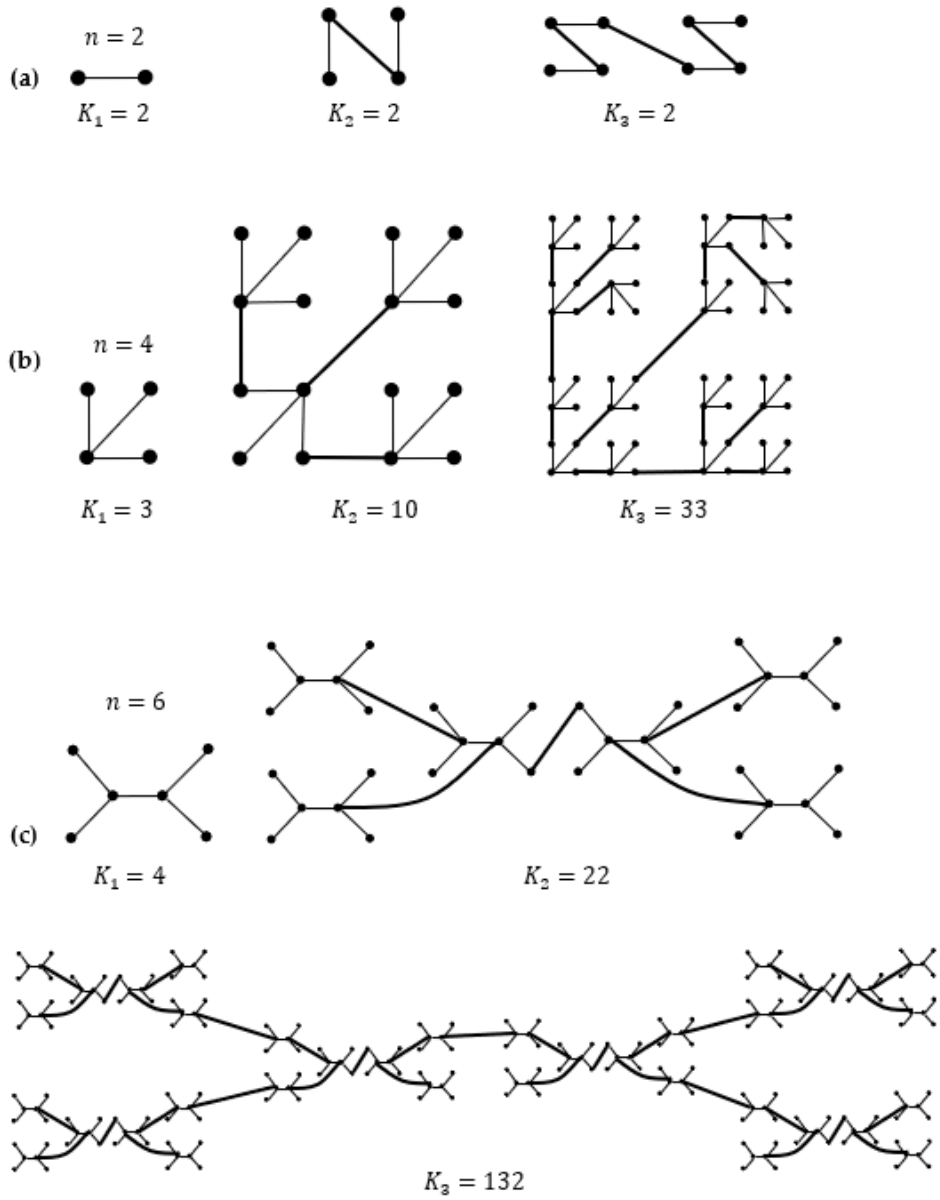


Figure 7. Spanning trees of prefractal graphs G_1, G_2, G_3 with leaf vertices.

2.4. Balanced Complete Bipartite Subgraph

INSTANCE: Bipartite graph $G = (V, E)$, positive integer $K \leq |V|$.

QUESTION: Are there two disjoint subsets $V_1, V_2 \subseteq V$ such that $|V_1| = |V_2| = K$ and such $u \in V_1$, $v \in V_2$ implies that $\{u, v\} \in E$?

Theorem 7. *A complete bipartite subgraph exists only on the seed for any prefractal graph.*

Proof of Theorem 7. The complete bipartite subgraph $G_1' = (V_1^1, V_1^2, E_1')$ of graph G_1 matches with the complete bipartite subgraph $H' = (W^1, W^2, Q')$ of the seed H according to the conditions of the theorem.

Next, we consider the case of generating a prefractal graph while preserving the adjacency of old edges. Spanning complete bipartite subgraphs are selected in the seeds (including all vertices) of G_2. Otherwise, the subgraph of G_2 may split into separate components. Let one vertex $w' \in W^1$ be incident to the old edges of G_1' or, in another way, w' also belongs to the part V_1^1 (see Figure 8). Then, to preserve the bipartition, set W^1 is included in V_1^1: $W^1 \subset V_1^1$. The second set W^2 is included in V_1^2: $W^2 \subset V_1^2$. Since

the adjacence of the old edges is preserved, vertices of W^2 are not adjacent to vertices V_1^1 (except vertices W^1). Then the definition of completeness of a bipartite graph is violated, where each vertex of one part must be adjacent to each vertex of the second part. Thus, the selection of a complete bipartite subgraph of any seed in G_2 does not allow the formation of a complete bipartite subgraph G_2' in G_2. Adhering to the proof of violation of the completeness in G_l, $l = 1, 2, \ldots, L$ it is impossible to select a spanning complete bipartite subgraph G_l'. □

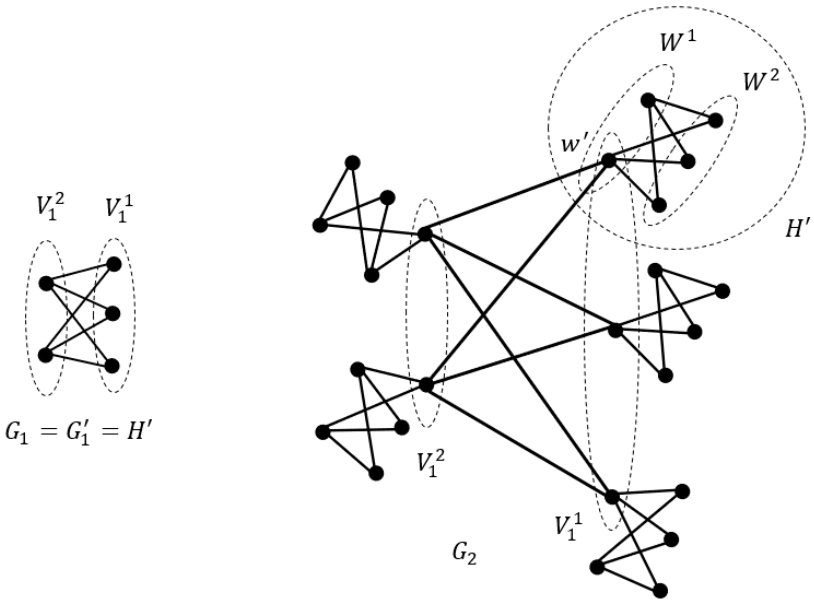


Figure 8. Prefractal graph G_2 generated by a complete bipartite seed, preserving the adjacency of old edges.

Next, we consider the case of generating a prefractal graph, the adjacency of the old edges of which is not preserved. The old edges e_1 and e_2 the adjacent ones in G_1' are not adjacent in G_2 (see Figure 9). The ends w_1' and w_2' of the old edges belong to the same part W^2. Otherwise, the condition of bipartition is violated, and the vertices w_1', w_2' will be adjacent. The second end w_2'' of the old edge e_2 in a bipartite seed should be adjacent to w_1'. However, such an edge does not exist since the old edges are not adjacent, and the new edges are present only in the seeds. Thus, the condition of completeness of the bipartite graph is violated. Thus, when the adjacency of the old edges in G_2 is not preserved, it is impossible to form a complete bipartite subgraph G_2'. Following the proof of violation of the completeness, it is impossible to select a spanning complete bipartite subgraph G_l', $l = 1, 2, \ldots, L$.

For each graph G_l with an arbitrary adjacency of old edges, it is possible to select a complete bipartite subgraph separately on each seed of rank $l = 1, 2, \ldots, L$. It corresponds to generating a prefractal graph by a seed, and is thus a complete bipartite graph.

If the adjacency of the old edges is preserved, a complete bipartite subgraph can be selected only in a single seed of any rank, and in the case when the old edges are not adjacent, only in a single seed of rank L.

Consequence 9. *A balanced complete bipartite subgraph exists only in a single seed of any prefractal graph.*

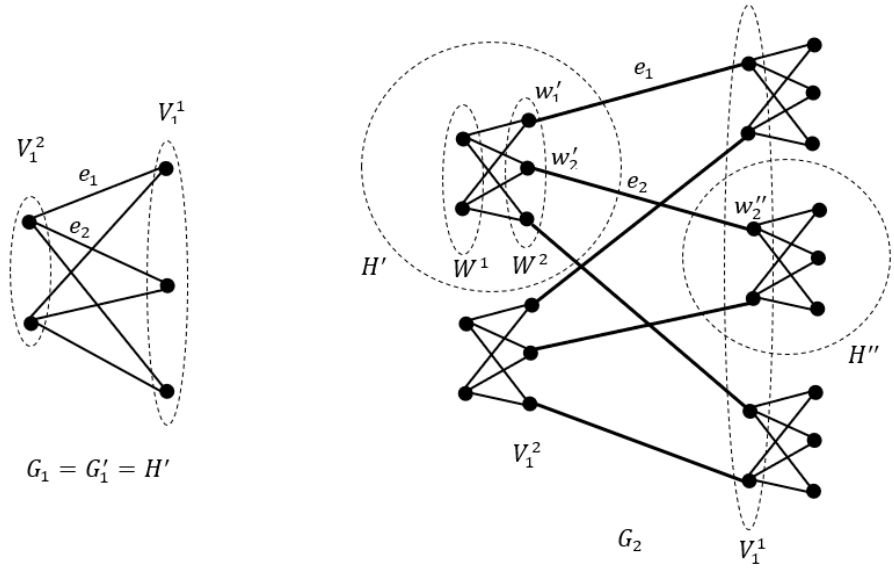


Figure 9. Prefractal graph G_2 generated by a complete bipartite seed; old edges are not adjacent.

2.5. Bipartite Subgraph

INSTANCE: Graph $G = (V, E)$, positive integer $K \leq |E|$.

QUESTION: Is there a subset $E' \subseteq E$ such that $G' = (V, E')$ is bipartite?

An algorithm for selecting a bipartite subgraph is proposed to formulate and prove the theorem. Let prefractal graph G_L be generated with preserving the adjacency of the old edges, and there is a spanning bipartite subgraph on the seed. The following is a description of an algorithm for packing a prefractal graph into a bipartite graph.

Algorithm 3 can be immediately applied to a prefractal graph generated by a bipartite seed. Any known algorithm for selecting a bipartite subgraph can be used as a procedure.

Algorithm 3 Algorithm for Packing a Bipartite Graph

Input: prefractal graph $G_L = (V_L, E_L)$.
Output: spanning bipartite subgraph $G_L' = (V_L^1, V_L^2, E_L')$ of G_L.
1. Select a bipartite subgraph $G_1' = (V_1^1, V_1^2, E_1')$ of G_1 using the procedure. This action corresponds to the selection of a bipartite subgraph $H' = (W^1, W^2, Q')$ of H.
for $l = 2$ to L do:
for $s = 1$ to n^{l-1} do:
l.s. Select a bipartite subgraph $H_s^{(l)} = \left(W_{l,s}^1, W_{l,s}^2, Q_{l,s}\right)$ of seed $z_s^{(l)}$ using the procedure.
$G_l' = (V_l^1, V_l^2, E_l')$ is formed as follows: one part $W_{l,s}^1$ is added to the part of G_{l-1}' with which it intersects; the second part $W_{l,s}^2$ is added to the part of G_{l-1}' that has no common vertices.
If $V_l^1 \cap W_{l,s}^1 \neq \varnothing$ then $V_l^1 = V_l^1 + W_{l,s}^1$ and $V_l^2 = V_l^2 + W_{l,s}^2$.
Else $V_l^1 \cap W_{l,s}^2 \neq \varnothing$ then $V_l^2 = V_l^2 + W_{l,s}^1$ and $V_l^1 = V_l^1 + W_{l,s}^2$.
$L + 1$. At the output of step L, spanning bipartite subgraphs $G_1', G_2', \ldots, G_L'$ are selected, which corresponds to the selection of a bipartite subgraph of the prefractal graph G_L.
Procedure for Selecting a Spanning Bipartite Subgraph
Input: graph $G = (V, E)$.
Output: spanning bipartite subgraph $G' = (V^1, V^2, E')$ of G.

Figure 10 shows subgraphs G_1', G_2', G_3' that correspond to prefractal graphs generated by a bipartite seed. G_1' is a bipartite subgraph H'. Algorithm 3 for packing bipartite graphs G_2', G_3' is applied.

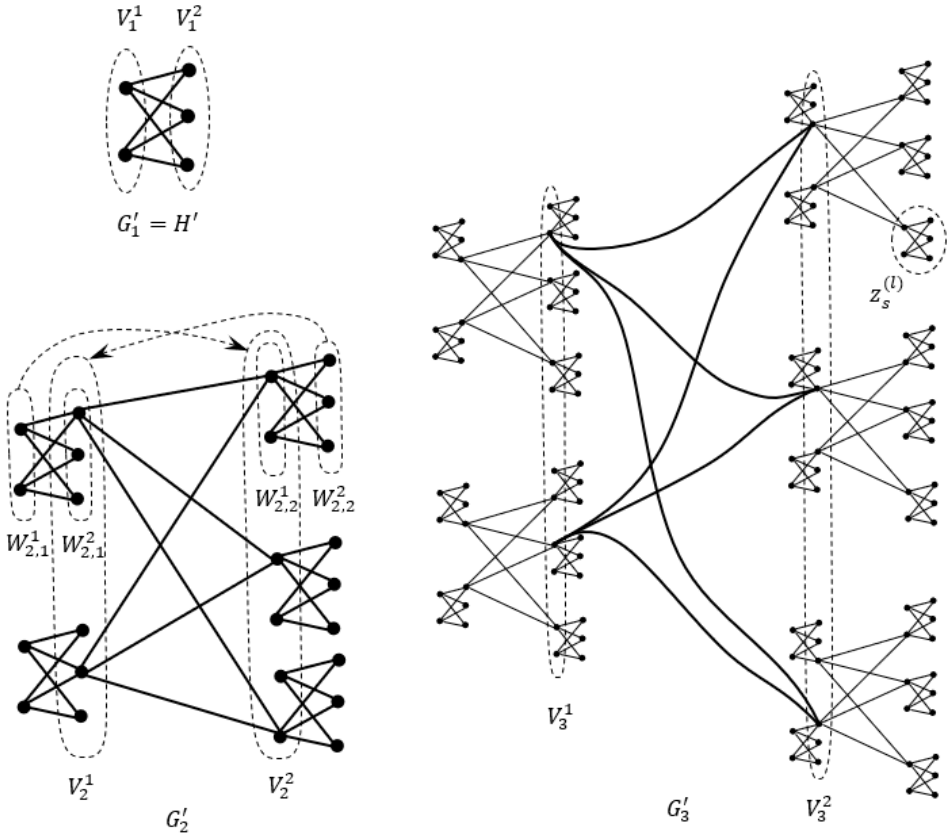


Figure 10. Prefractal graph G_3' generated by a complete bipartite seed, preserving the adjacency of old edges.

Theorem 8. *If there is a bipartite subgraph* $H' = (W^1, W^2, Q')$ *for the seed* H *and the adjacency of old edges is preserved, then there is a bipartite subgraph* $G_l' = (V_l^1, V_l^2, E_l')$ *for the prefractal graph* G_l, $l = 1, 2, \ldots, L$ *with* $|E_l'| = K$ *where* $K = k \cdot \left(n^l - 1\right)/(n-1)$, $k = |Q'|$.

Proof of Theorem 8. There is a bipartite subgraph $G_1' = (V_1^1, V_1^2, E_1')$ of G_1 under the condition of existence of a bipartite subgraph $H' = (W^1, W^2, Q')$ of seed H and $|E_l'| = |Q'| = k$. Next, graph G_2 is considered. Bipartite subgraph $H_1^{(1)} = \left(W_{1,1}^1, W_{1,1}^2, Q_{1,1}\right)$ is selected in seed $z_1^{(1)}$ (corresponding to G_1'). Vertices $W_{1,1}^1$ are added to one part V_2^1 and vertices $W_{1,1}^2$ to another V_2^2. $H_1^{(2)} = \left(W_{2,1}^1, W_{2,1}^2, Q_{2,1}\right)$ in $z_s^{(l)}$ is selected. One part $W_{2,1}^1$ is added to the part of G_1' with which it intersects; the second part $W_{2,1}^2$ is added to the part of G_1' that has no common vertices:

if $V_2^1 \cap W_{2,1}^1 \neq \varnothing$ then $V_2^1 = V_2^1 + W_{2,1}^1$ and $V_2^2 = V_2^2 + W_{2,1}^2$;
else $V_2^1 \cap W_{2,1}^2 \neq \varnothing$ then $V_2^2 = V_2^2 + W_{2,1}^1$ and $V_2^1 = V_2^1 + W_{2,1}^2$.

Since the adjacency of the old edges is preserved, then $H_1^{(2)}$ intersects G_1' in a common vertex. Adding the parts $W_{2,1}^1, W_{2,1}^2$ to the corresponding parts V_2^1, V_2^2 does not violate the definition of a bipartite graph. The total number of edges $H_1^{(2)}$ and G_1' is equal to $|E_1'| + |Q_{2,1}| = k + k = 2k$. The selection of alternately bipartite subgraphs $H_s^{(2)}$ in the remaining seeds $z_s^{(2)}$ and the addition of parts $W_{2,s}^1, W_{2,s}^2$ to V_2^1, V_2^2 in the same way corresponds to the selection of a bipartite subgraph G_2' in G_2. Furthermore, since G_2 contains one seed of the first rank and n seeds of the second rank, $|E_2'| = k \cdot |Q'| = k \cdot (n+1)$. Next, $G_3, G_4, \ldots, G_L$ are sequentially considered, for which $G_3', G_4', \ldots, G_L'$ are selected in accordance with the described approach. Adding bipartite subgraphs H' of seeds H does not

violate the definition of a bipartite graph. All vertices within one part are not adjacent, and adjacency exists only between vertices of two different parts.

At each step $G_1, G_2, \ldots, G_L$, n^{l-1} seeds are added; that is, there entire $1 + n + n^2 + \ldots + n^{l-1} = \left(n^l - 1\right)/(n-1)$ seeds in G_L. Thus, there is a bipartite subgraph $G'_l = \left(V_l^1, V_l^2, E'_l\right)$ of G_l, $l = 1, 2, \ldots, L$, with $\left|E'_l\right| = k \cdot \left(n^l - 1\right)/(n-1) = K$. □

Consequence 10. *The proof of Theorem 8 is the justification of Algorithm 3.*

2.6. Planar Subgraph

INSTANCE: Graph $G = (V, E)$, positive integer $K \leq |E|$.
QUESTION: Is there a subset $E' \subseteq E$ with $|E'| \geq K$ such that $G' = (V, E')$ is planar?

Lemma 9. *The union of two planar graphs at one common vertex is also a planar graph.*

Proof of Lemma 9. When two planar graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are united at one common vertex w^* no new edges appear in $G^* = G_1 \cup G_2$ (see Figure 11). □

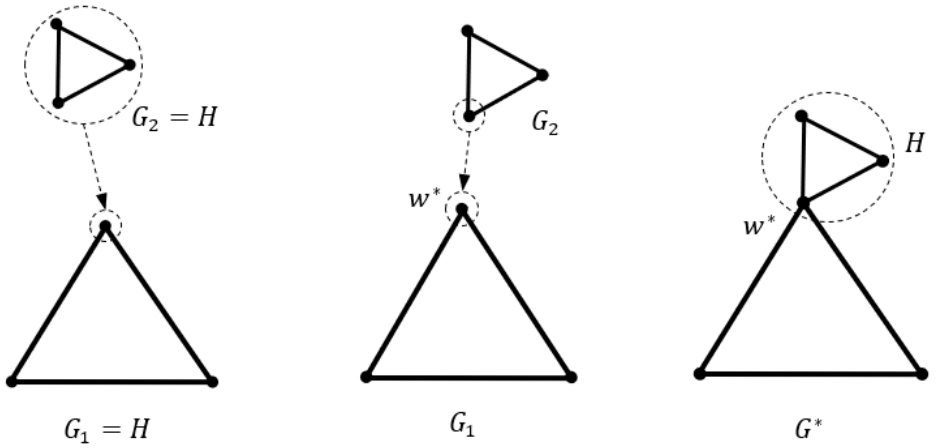


Figure 11. Operation of replacing a vertex with a seed or union of a graph G_1 and a seed H in one vertex.

The operation of replacing a vertex with a seed (RVW) while preserving the old edges' adjacency corresponds to the union of two graphs $G_1 = (V_1, E_1)$ and $H = (W, Q)$ at one common vertex in a new graph $G^* = (V^*, E^*)$: $V^* = (V_1 \cup W) - \{w^*\}$, $E^* = (E_1 \cup Q)$, or, in another way, joining a seed H. Under the condition of planarity H, the new graph G^* is also planar.

Consequence 10. *When two arbitrary graphs are united at one common vertex, their planar subgraphs are united into a planar graph.*

Theorem 10. *If there is a planar subgraph $H' = (W, Q')$ in seed $H = (W, Q)$, and the adjacency of old edges is preserved, then there is a planar subgraph $G'_l = \left(V, E'_l\right)$ in prefractal graph G_l, $l = 1, 2, \ldots, L$, for which $\left|E'_l\right| = K$, where $K = k \cdot \left(n^l - 1\right)/(n-1)$, $k = |Q'|$.*

Proof of Theorem 10. There is a planar subgraph G'_1 in G_l under the conditions for the existence of a planar subgraph H' in seed H, where $\left|E'_1\right| = |H'| = k$. Since the adjacency of the old edges is preserved, for constructing the prefractal graph G_l, $l = 2, 3, \ldots, L$ a seed H is alternately attached to each vertex of G_{l-1}. G'_{l-1} is planar, and joining planar subgraphs H' to its vertices does not violate the planarity of G'_l following Lemma 9 and Consequence 10. At each step $l = 1, 2, \ldots, L$, n^{l-1} seeds are added, that is, total $1 + n + n^2 + \ldots + n^{l-1} = (n^l - 1)/(n-1)$ seeds in G_l.

Thus, there is a planar subgraph $G'_l = (V, E'_l)$ in prefractal graph G_l, for which $|E'_l| = k \cdot \left(n^l - 1\right)/(n-1)$.

Figure 12 shows the procedure for isolating a planar subgraph of a prefractal graph. Following Theorem 10, a planar subgraph H' is selected in the seed H (a), and a planar subgraph G'_2 in G_2 (b), where $|E_2| = 10 \cdot \frac{(5^2-1)}{5-1} = 10 \cdot 24/4 = 60$ and $K = |E'_2| = 9 \cdot \frac{(5^2-1)}{5-1} = 9 \cdot 24/4 = 54$.

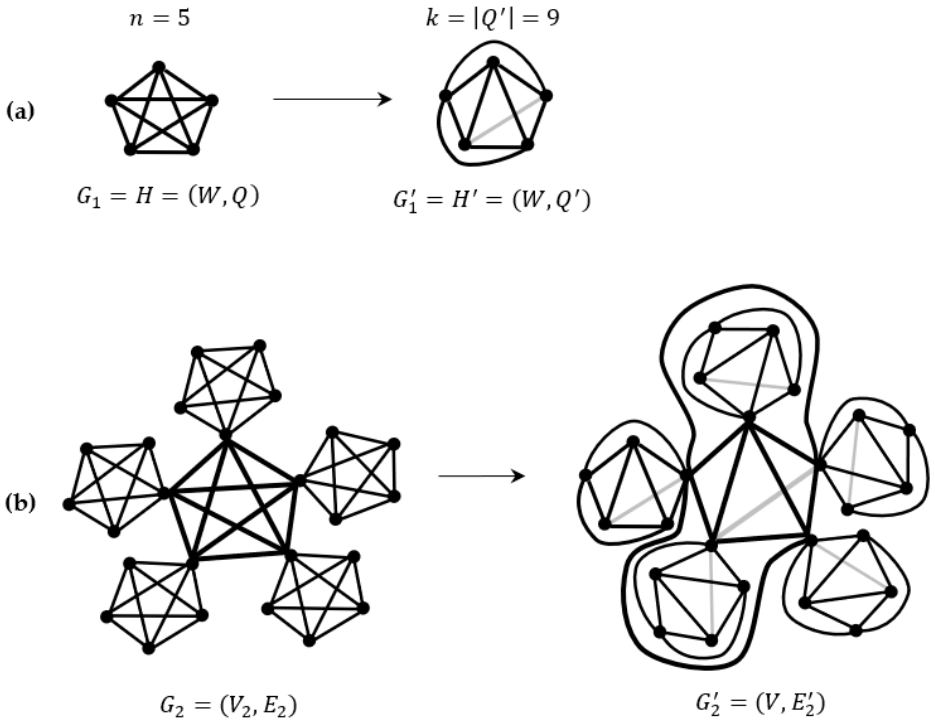


Figure 12. Procedure for isolating a planar subgraph of a prefractal graph when the adjacency of old edges is preserved.

3. Results and Discussion

In this paper, we have discussed six known NP-complete problems in relation to the class of prefractal graphs. For each of the problems, conditions are proposed under which we can answer the question positively or negatively. Algorithms for finding a solution are proposed for some problems.

In the problem of the isomorphism of a subgraph, an algorithm for numbering the vertices of a prefractal graph and a figure with an example are proposed. A theorem is also proposed for the isomorphism of two prefractal graphs, provided that the adjacency of the old edges is preserved.

In the second problem, on a degree constrained spanning tree, an algorithm for finding a spanning tree of minimum weight (MST) of any prefractal graph is proposed. The complexity of the algorithm is shown, where any known MST algorithm, including the modern one, can be used as a procedure. The procedure is applied to the seeds, and then the subgraphs are united into the MST of the entire prefractal graph. The execution time of the algorithm is linear $O(c \cdot N)$ and depends on the number of vertices $N = n^l$. The parameter $c = 4n^2$ is a constant since the number of seed vertices n is fixed before building the prefractal graph. In the theorem, conditions are proposed under which there exists a spanning tree with vertex degrees at most $K = k + 1$, where the number k is the maximum vertex degree of the generating seed.

In the problem of a maximum leaf spanning tree, a lemma is proposed that any leaf vertex is on a seed of the last rank. This property is associated with the process of constructing a prefractal graph when at each step, the vertices are replaced by seeds. As

for other problems, the characteristics of the entire prefractal graph depend on the seed. This dependence is key and requires additional study.

For the problem of a balanced complete bipartite subgraph, we have proved the theorem that it can be distinguished only on one separate seed. The rule for constructing a prefractal graph preserves this property throughout the entire trajectory. As one of the conclusions, we can suggest the impossibility of identifying a balanced full bipartite subgraph, for example, in social networks. This claim requires research and additional evidence.

For the problem of selecting a bipartite subgraph, a packing algorithm is proposed in the case of preserving the adjacency of old edges. These results can be reformulated as constructing a graph with given characteristics. That is, how can one construct a graph so that a bipartite subgraph exists in it? For such a constructed graph, a packing algorithm can be applied.

The problem of the planarity of a prefractal graph is also related to the design of graphs with predetermined characteristics. If the seed is a planar graph and the adjacency of the old edges is preserved, then for the entire sequence $l = 1, 2, \ldots, L$ the graphs G_l are planar.

As one of the results, we can single out the possibility of developing algorithms for solving optimization problems with polynomial complexity. Such a modular execution of sequential algorithms on subgraphs (seeds) will allow them to be parallelized in the future. Furthermore, the use of well-known algorithms in the form of procedures will allow researchers in the future to change them to new ones with improved characteristics.

It is assumed that the study of individual NP-complete problems on prefractal graphs will make it possible to form a set of rules for identifying solvability conditions for any known NP-complete problem in this class of graphs, and also to approve the class of prefractal graphs as a special class for which there are conditions for the solvability of NP-complete problems. An example of this is demonstrated in [51] for inherited classes.

Therefore, for all theorems, consequences are proposed with a generalization of the results to the sequence $G_1, G_2, \ldots, G_L$. Since G_L is a dynamic graph that changes in the trajectory $l = 1, 2, \ldots, L$ and for each G_l an algorithm and a solution to the problem are proposed. In further studies, we plan to pay more attention to the formulation of problems for dynamic graphs, to definitions of solutions to dynamic problems and to the study of the characteristics and properties of prefractal graphs as a subclass of dynamic graphs.

4. Conclusions

The paper considers and investigates NP-complete problems for one of the classes of dynamic graphs—prefractal graphs. For each of the problems, solvability conditions are given under which it is possible to obtain an existing answer for some subproblems, as well as to construct polynomial algorithms for finding solutions for some. A number of corollaries generalize the results obtained, or provide conditions for the absence of a solution.

In this paper, we study only a small part of the known NP-complete problems. The study of the entire set of problems and the selection of solvability conditions for this class of dynamic graphs—prefractal graphs—will essentially allow us to talk about the selection of an entire subclass of graphs with conditions for the solvability of NP-complete problems.

The development of polynomial algorithms for dynamic graphs will make it possible to solve with improved characteristics such well-known applied problems as the selection of subgraphs in large dynamic networks [52,53]; the detection of communities in social networks [54–56]; the solution of multicriteria problems in large-scale transport and logistics systems [57]; the searching and highlighting of DDoS attacks in cryptocurrency systems [58,59] and many other problems.

In the future, we plan to introduce definitions for a dynamic prefractal graph, setting a multicriteria graph-theoretic problem, solving a dynamic problem and the concept of

topological time. Furthermore, we also aim to define families of dynamic problems for prefractal graphs.

The number of applied problems in large networks is only growing. With the development of the direction of big data analysis, the increase in the number of subscribers and the complication of networks, the study of problems on dynamic graphs of large dimensions is becoming more and more urgent. Thus, we propose to use all the available tools of dynamic prefractal graphs to solve optimization problems in large networks, as well as to design networks with given characteristics, including predicting the development of real networks [60,61].

It should be noted separately that it is necessary to define fractal graphs and related concepts. Since a fractal graph is an infinite object, what does the formulation of the optimization problem look like, how do the algorithms work and what is the solution? For this purpose, we plan to publish a separate article on fractal and prefractal graphs to familiarize the scientific community with this class of special graphs and to consecrate their properties and characteristics.

Funding: This research was funded by Financial University under the Government of the Russian Federation.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W.H. Freeman and Company: San Francisco, CA, USA, 1979.
2. Garey, M.R.; Johnson, D.S.; Stockmeyer, L. Some Simplified NP-Complete Problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing (STOC '74), Seattle, WA, USA, 30 April–2 May 1974*; Association for Computing Machinery: New York, NY, USA, 1974; pp. 47–63. [CrossRef]
3. Murty, K.G.; Kabadi, S.N. Some NP-complete problems in quadratic and nonlinear programming. *Math. Program.* **1987**, *39*, 117–129. [CrossRef]
4. Gavril, F. Some NP-Complete Problems on Graphs. In *The 1977 Conference on Information Sciences and Systems, The Johns Hopkins University, Baltimore, MD, USA*; CS Technion report CS-2011-05; Computer Science Department, Technion: Haifa, Isarael, 2011; pp. 91–95. Available online: http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2011/CS/CS-2011-05.pdf (accessed on 28 October 2021).
5. Karci, A. Finding Innovative and Efficient Solutions to NP-Hard and NP-Complete Problems in Graph Theory. *Comput. Sci.* **2020**, *5*, 137–143. Available online: https://dergipark.org.tr/en/pub/bbd/issue/57870/715712 (accessed on 5 August 2021).
6. Kasarkin, A.V.; Levin, I.I.; Sorokin, D.A. New iteration parallel-based method for solving graph NP-complete problems with reconfigurable computer systems. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *919*, 052007. [CrossRef]
7. Huh, J.-H.; Hwa, J.; Seo, Y.-S. Hierarchical System Decomposition Using Genetic Algorithm for Future Sustainable Computing. *Sustainability* **2020**, *12*, 2177. [CrossRef]
8. Dantas, S.; De Figueiredo, C.M.H.; Petito, P.; Teixeira, R.B. A General Method for Forbidden Induced Subgraph Sandwich Problem NP-completeness. *Electron. Notes Theor. Comput. Sci.* **2019**, *346*, 393–400. [CrossRef]
9. Lacroix, M.; Mahjoub, A.R.; Martin, S.; Picouleau, C. On the NP-completeness of the perfect matching free subgraph problem. *Theor. Comput. Sci.* **2012**, *423*, 25–29. [CrossRef]
10. Asdre, K.; Nikolopoulos, S.D. NP-completeness results for some problems on subclasses of bipartite and chordal graphs. *Theor. Comput. Sci.* **2007**, *381*, 248–259. [CrossRef]
11. Martínez-Pérez, I.M.; Zimmermann, K.-H. Parallel bioinspired algorithms for NP complete graph problems. *J. Parallel Distrib. Comput.* **2009**, *69*, 221–229. [CrossRef]
12. Ohya, M.; Volovich, I.V. New quantum algorithm for studying NP-complete problems. *Rep. Math. Phys.* **2003**, *52*, 25–33. [CrossRef]
13. Kulmburg, A.; Althoff, M. On the co-NP-completeness of the zonotope containment problem. *Eur. J. Control* **2021**, in press. [CrossRef]
14. Komjáthy, J.; Simon, K. Generating hierarchial scale-free graphs from fractals. *Chaos Solitons Fractals* **2011**, *44*, 651–666. [CrossRef]
15. Moreno-Pulido, S.; Pavón-Domínguez, P.; Burgos-Pintos, P. Temporal evolution of multifractality in the Madrid Metro subway network. *Chaos Solitons Fractals* **2021**, *142*, 110370. [CrossRef]

16. Criado-Alonso, Á.; Battaner-Moro, E.; Aleja, D.; Romance, M.; Criado, R. Enriched line graph: A new structure for searching language collocations. *Chaos Solitons Fractals* **2021**, *142*, 110509. [CrossRef]
17. Teufl, E.; Wagner, S. Enumeration of matchings in families of self-similar graphs. *Discret. Appl. Math.* **2010**, *158*, 1524–1535. [CrossRef]
18. Gong, H.; Jin, X. A general method for computing Tutte polynomials of self-similar graphs. *Phys. A Stat. Mech. Its Appl.* **2017**, *483*, 117–129. [CrossRef]
19. Dorogov, A.Y. Morphological model of self-similar multilayer neural networks. *Procedia Comput. Sci.* **2021**, *186*, 366–373. [CrossRef]
20. Kochkarov, A.A.; Kochkarov, R.A. Prefractal graphs in designing compound structures. *Keldysh Inst. Prepr.* **2003**, *10*, 1–21.
21. Perepelitsa, V.A.; Sergienko, I.V.; Kochkarov, A.M. Recognition of fractal graphs. *Cybern. Syst. Anal.* **1999**, *35*, 572–585. [CrossRef]
22. Zegzhda, P.D.; Ivanov, D.V.; Moskvin, D.A.; Ivanov, A.A. The Use of Adjacency Series for Recognition of Prefractal Graphs in Assessing VANET Cybersecurity. *Autom. Control Comput. Sci.* **2018**, *52*, 901–905. [CrossRef]
23. Malinetskii, G.G.; Kochkarov, A.A.; Kochkarov, R.A. Issues of dynamic graph theory. *Comput. Math. Math. Phys.* **2015**, *55*, 1590–1596.
24. Gignoux, J.; Chérel, G.; Davies, I.D.; Flint, S.R.; Lateltin, E. Emergence and complex systems: The contribution of dynamic graph theory. *Ecol. Complex.* **2017**, *31*, 34–49. [CrossRef]
25. Peng, H.; Du, B.; Liu, M.; Liu, M.; Ji, S.; Wang, S.; Zhang, X.; He, L. Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning. *Inf. Sci.* **2021**, *578*, 401–416. [CrossRef]
26. Kochkarov, A.A.; Kochkarov, R.A. Parallel algorithm for finding the shortest path on a prefractal graph. *Comput. Math. Math. Phys.* **2004**, *44*, 1157–1162.
27. Kochkarov, A.A.; Kalashnikov, N.V.; Kochkarov, R.A. Identifying Bots in Social Networks Using the Example of LiveJournal. *World New Econ.* **2020**, *14*, 44–50. [CrossRef]
28. Impedovo, A.; Loglisci, C.; Ceci, M.; Malerba, D. Condensed representations of changes in dynamic graphs through emerging subgraph mining. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103830. [CrossRef]
29. Cheong, S.-H.; Si, Y.-W.; Wong, R.K. Online force-directed algorithms for visualization of dynamic graphs. *Inf. Sci.* **2021**, *556*, 223–255. [CrossRef]
30. Zhang, Z.; Li, Y.; Song, H.; Dong, H. Multiple dynamic graph based traffic speed prediction method. *Neurocomputing* **2021**, *461*, 109–117. [CrossRef]
31. Li, W.; Xiao, D. Dimensions Of Measure On General Sierpinski Carpet. *Acta Math. Sci.* **1999**, *19*, 81–85. [CrossRef]
32. Teplyaev, A. Spectral Analysis on Infinite Sierpiński Gaskets. *J. Funct. Anal.* **1998**, *159*, 537–567. [CrossRef]
33. Strichartz, R.S. Taylor Approximations on Sierpinski Gasket Type Fractals. *J. Funct. Anal.* **2000**, *174*, 76–127. [CrossRef]
34. Jakovac, M.; Klavžar, S. Vertex-, edge-, and total-colorings of Sierpiński-like graphs. *Discret. Math.* **2009**, *309*, 1548–1556. [CrossRef]
35. Guan, J.; Wu, Y.; Zhang, Z.; Zhou, S.; Wu, Y. A unified model for Sierpinski networks with scale-free scaling and small-world effect. *Phys. A Stat. Mech. Its Appl.* **2009**, *388*, 2571–2578. [CrossRef]
36. Klavžar, S.; Peterin, I.; Zemljič, S.S. Hamming dimension of a graph—The case of Sierpiński graphs. *Eur. J. Comb.* **2013**, *34*, 460–473. [CrossRef]
37. Hinz, A.M.; Klavžar, S.; Zemljič, S.S. A survey and classification of Sierpiński-type graphs. *Discret. Appl. Math.* **2017**, *217*, 565–600. [CrossRef]
38. Estrada-Moreno, A.; Rodríguez-Velázquez, J.A. On the General Randić index of polymeric networks modelled by generalized Sierpiński graphs. *Discret. Appl. Math.* **2019**, *263*, 140–151. [CrossRef]
39. Xue, B.; Zuo, L.; Wang, G.; Li, G. Shortest paths in Sierpiński graphs. *Discret. Appl. Math.* **2014**, *162*, 314–321. [CrossRef]
40. Xue, B.; Zuo, L.; Li, G. The hamiltonicity and path t-coloring of Sierpiński-like graphs. *Discret. Appl. Math.* **2012**, *160*, 1822–1836. [CrossRef]
41. Hinz, A.M.; Holz auf der Heide, C. An efficient algorithm to determine all shortest paths in Sierpiński graphs. *Discret. Appl. Math.* **2014**, *177*, 111–120. [CrossRef]
42. Harary, F. *Graph Theory*; Addison-Wesley Pub. Co.: Boston, MA, USA, 1969.
43. Iordanskii, M.A. A Constructive Classification of Graphs. *Modeling Anal. Inf. Syst.* **2012**, *19*, 144–153. [CrossRef]
44. Kochkarov, A.M. *Recognition of Fractal Graphs. Algorithmic Approach*; RAS SAO: Nizhny Arkhyz, Russia, 1998.
45. Kochkarov, R.A. *Problems of Multicriteria Optimization on Multi-Weighted Prefractal Graphs*; Akademinnovatsiya: Moscow, Russia, 2014; ISBN 978-5-906761-01-9.
46. Aziz, F.; Akbar, M.S.; Jawad, M.; Malik, A.H.; Uddin, M.I.; Gkoutos, G.V. Graph characterisation using graphlet-based entropies. *Pattern Recognit. Lett.* **2021**, *147*, 100–107. [CrossRef]
47. Aziz, F.; Ullah, A.; Shah, F. Feature selection and learning for graphlet kernel. *Pattern Recognit. Lett.* **2020**, *136*, 63–70. [CrossRef]
48. Moreno, L.Q. Graphlets and Motifs in Biological Networks. *Encycl. Bioinform. Comput. Biol.* **2019**, *2*, 814–820. [CrossRef]
49. Ahmed, N.K.; Neville, J.; Rossi, R.A.; Duffield, N. Efficient Graphlet Counting for Large Networks. In Proceedings of the IEEE International Conference on Data Mining (ICDM), Atlantic City, NJ, USA, 14–17 November 2015; pp. 1–10. [CrossRef]
50. Melrose, J. Hierarchical Fractal Graphs And Walks Thereupon. In *Fractals in Physics*; Pietronero, L., Tosatti, E., Eds.; Elsevier: Amsterdam, The Netherlands, 1986; pp. 365–368. [CrossRef]

51. Alekseev, V.E. On easy and hard hereditary classes of graphs with respect to the independent set problem. *Discret. Appl. Math.* **2003**, *132*, 17–26. [CrossRef]
52. Patra, S. Discovery of network motifs based on induced subgraphs using a dynamic expansion tree. *Comput. Biol. Chem.* **2021**, *93*, 107530. [CrossRef]
53. Zhang, Z.; Chen, D.; Wang, J.; Bai, L.; Hancock, E.R. Quantum-based subgraph convolutional neural networks. *Pattern Recognit.* **2019**, *88*, 38–49. [CrossRef]
54. Adraoui, M.; Retbi, A.; Idrissi, M.K.; Bennani, S. Maximal cliques based method for detecting and evaluating learning communities in social networks. *Future Gener. Comput. Syst.* **2022**, *126*, 1–14. [CrossRef]
55. Zhao, X.; Liang, J.; Wang, J. A community detection algorithm based on graph compression for large-scale social networks. *Inf. Sci.* **2021**, *551*, 358–372. [CrossRef]
56. Mishra, S.; Singh, S.S.; Mishra, S.; Biswas, B. TCD2: Tree-based community detection in dynamic social networks. *Expert Syst. Appl.* **2021**, *169*, 114493. [CrossRef]
57. Yatskin, D.V.; Kochkarov, A.A.; Kochkarov, R.A. Modeling of Transport and Logistics Systems and the Study of the Structural Stability. *Manag. Sci. Russ.* **2020**, *10*, 102–111. [CrossRef]
58. Kochkarov, A.A.; Osipovich, S.D.; Kochkarov, R.A. Recognizing DDoS attacks on the bitcoin cryptocurrency system. In Proceedings of the 2019 Symposium on Cybersecurity of the Digital Economy (CDE'19), Kazan, Russia, 22–24 May 2019; pp. 220–223.
59. Kochkarov, A.A.; Osipovich, S.D.; Kochkarov, R.A. DDoS attacks recognition technique on cryptocurrency systems. *Prot. Inf. Inside* **2020**, *2*, 32–35.
60. Su, Y.; Zhou, K.; Zhang, X.; Cheng, R.; Zheng, C. A parallel multi-objective evolutionary algorithm for community detection in large-scale complex networks. *Inf. Sci.* **2021**, *576*, 374–392. [CrossRef]
61. Hwangbo, S.; Heo, S.; Yoo, C. Development of deterministic-stochastic model to integrate variable renewable energy-driven electricity and large-scale utility networks: Towards decarbonization petrochemical industry. *Energy* **2022**, *238*, 122006. [CrossRef]