Tensor Models of Fractal Graphs for Elastic Networks

A. S. Semenov

Moscow Aviation Institute (National Research University), Moscow, Russia

Abstract.The article presents the results of research on fractal (self-similar) graphs in relation to elastic computing. A characteristic feature of such graphs is their ability to unfold (increase dimensionality) and fold (decrease dimensionality). Two approaches to forming fractal graphs are considered: based on Kronecker product and fractal algebra. The interrelationship of algebraic operations of forming fractal graphs (linear graphs, grids, hypercubes, and trees) with tensor operations and tensor representation based on the integration of adjacency matrices and event vectors of elastic systems is presented. Definitions of corresponding types of dynamically changing tensors are introduced. An analysis of the properties of elastic fractal graphs and related tensor models is conducted

Keywords: Kronecker graphs, fractal graphs, fractal algebra, elastic networks, tensor models.

DOI 10.14357/20718632230412 **EDN** SSMYLI

Introduction

The emergence of self-organizing, reconfigurable, and elastic network systems has created a problem for modelling network transformation. To be able to transform, the system's organization must be elastic [1]. Such a transformation should be modeled during the design stage and taken into account throughout operations, allowing the system's life cycle to be extended. The development of mathematical models for the analysis of elastic systems is related to the construction of models that adequately simulate it behavior. An adequate technique to model network systems is to represent them by using dynamically transformable graphs. The qualitative properties of graph-based models gain in value as systems and their dynamics get more complicated. This means that the internal development activity and the nature of intended behavior should be taken into consideration in models. The qualitative properties of graphs that are evident in self-organization modeling play a specific role in this case. The use of graphs will allow to restore the structure of the system automatically and qualitatively. Furthermore, a graph representation of a network system gives not only a visual representation, but also allows the application of algorithms already known.

It was observed that network patterns could be used to synthesize network systems with different topologies. Graphs synthesized and modified on the basis of isomorphic subgraphs (patterns) have been called fractal (self-similar) graphs [2, 3]. The ability to transform the structure over existing subgraphs, as well as the ability to represent models with smaller graphs and map on mathematical metrics of similarity [4] allows for a decreasing dimension of the solving task, are all qualitative properties of fractal graphs. Fractal graphs are synthesized by performing simple operations on initial graphs.

The Kronecker product [5] and fractal algebra are two methods for synthesizing fractal graphs. Thus, the Kronecker product is applicable to tensor computation for data analysis [6] and have been proven to be one of the most promising models for

real-world networks [7]. It is a technique for constructing self-similar adjacency matrices, resulting in a block matrix that can be visualized with graphs. Kronecker graphs are useful for theoretical analysis and proof because they contain many significant patterns of realistic networks. The model really fits the real networks. Furthermore, there exist efficient algorithms for locating graphs that fit key patterns in real networks. Many applications, such as graph compression, extrapolation, sampling, and data anonymization, can be constructed on top of Kronecker graphs. Despite this, they are still the subject of research.

Another way to construct fractal graphs is based on simple operations over the pattern described by the initial graph. Operations are systematized into fractal algebra [8, 9]. In the paper, algebraic algorithms for the synthesis of fractal graphs such as linear graphs, grids, hypercubes, and trees are introduced. The representation of adjacency matrices of these fractal graphs in the form of tensors [10] is developing as part of underlying algebraic method. Definitions of dynamically modifiable adjacency matrices [11] and the transformation principles that drive them are provided for these purposes [12–14]. This made it possible to form tensor models of fractal graphs by combining their adjacency matrices with the event vector of graph synthesis.

The paper is organized as follows. In the first section, both the fractal graph-based elastic model and its tensor presentation are introduced. The second section considers tensor models of elastic fractal graphs based on the Kronecker product and algebraic operations. The translation of the fractal algebra operations over graphs into tensor operations is given. In the third section, the properties of elastic fractal graphs and their associated tensor models are analyzed. Conclusions and contributions are presented in the later section.

1. Fractal Graph-based Dynamics

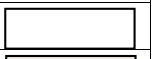
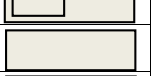
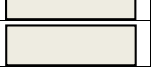
In this section, the fractal graph-based elasticity model and its tensor presentation are given. For matrices and vectors, the standard notation in bold italic format is used (e.g., $\boldsymbol{A}$, $\boldsymbol{K}$).

A. Fractal graph-based elasticity network model

The elastic concept involves transforming the system capacity on demand when the workload is changed. The incremental, decremental, and iterative nature of elastic networks directly impacts graph-based simulation and graph dynamics [15, 16]. The maximum workload that the system can handle in terms of request count is determined by the service's capacity. In order to achieve a balanced workload among the various vertices and control the system workload, the use of a fractal binary tree to control the system capacity can be considered. Let's look at a simplified example.

The initial elastic system capacity of 1,000 request per event is represented by a single vertex. To visualize the vertex a rectangle is used. The newly created vertices are denoted with white rectangles. At event 2, with an increase in workload to 2,000 requests per minute, it becomes necessary to increase the system capacity. A new vertex is nested in to the initial vertex, and its value is set to the capacity of 1,000 requests per minute. The total system capacity was increased to 2,000 requests per minute. Table 1 shows the workload that characterizes the data processed by a service's operations. The workload is based on fluctuating user demand over an 18-hour period, represented by events. Furthermore, the workload exhibited in Table 1 demonstrates fractal graph dynamics that could potentially affect the processing efficiency of the service's operations.

Table 1. Elastic system workload, events, and fractal graph dynamics

Event	*Workload*	*Graph*
1	1,000	
2	2,000	
3	3,000	
4	6,000	
5	7,000	
6	4,000	
7	2,000	
8	1,000	
9	10	

Further, all the events in the system from Table 1 are taken into account.

At event 3, an increase in workload to 3,000 requests per minute requires an increase in the system

capacity. A new vertex is nested into the initial vertex to the right.

At event 4, a new three vertex is nested into low level of the graph from left to right as a result of an increase in workload to 6,000 requests per minute.

At event 5, an increase in workload to 7,000 requests per minute was observed, necessitating another increase in capacity. There is currently an empty vertex on the right branch of the binary tree. A new vertex is nested to fill this empty vertex.

At event 6, the workload decreased to 4,000 requests per minute, allowing for a reduction in the system's capacity by removing 3 vertices from the lower level of the graph.

At event 7, a decrease in workload to 2,000 requests per minute allows for the removal of 2 vertices from the lower level of the graph, reducing the capacity to 2,000 requests per minute.

At event 8, a decrease in workload to 1,000 requests per minute allows for the removal of 1 vertex from the lower level of the graph, reducing the capacity to 1,000 requests per minute.

At event 9, a further decrease in workload is observed to 10 request per minute, rendering any changes to the system's capacity unnecessary.

It is possible to create different graphs corresponding to a variety of elasticity granularities by changing capacity units and strategies for changing system capacity.

The fractal graphs [17] dynamics of the networks satisfy the elasticity requirement.

Def.1. A graph G = (V,E) consists of a finite set of vertices V = $\{v_1,v_2,...,v_n\}$ and a finite set of edges E=$\{e_1,e_2,...,e_m\}$, for each edge $e_i \in E$ is unordered pairs of vertices $e_i = (v_i,v_j)$.

The number of vertices |V| defines the order of a graph, and the number of edges |E| defines the size of a graph.

Def.2. A graph $g_0 = (V_0, A_0)$ is a subgraph of G = (V,A) if $V_0 \subseteq V$ and every edge of g_0 is also an edge of G.

The subgraph g_0 (not a null subgraph) of a graph G is a component of G in two cases:

- if g_0 is a single isolated vertex of G
- or if g_0 is an induced subgraph, i.e., g_0 is formed from a subset of the vertices and edges of the graph G and any two vertices are connected to each other by paths.

Def 3. The simplest fractal graph-based model G_d consists of the initial component g_0 and the iterated function f:

$$G_d = f(f(...f(g_0))) = f_d(g_0), \text{ where} \quad (1)$$

f: G → G is an iterated function d = 1,2,...,n times, is written as follows:

g_0 is the initial component,
$g_1 = f(g_0)$
$g_2 = f(g_1)$
. . .
$G_d = f(g_{d-1})$

The result of the iteration function f is a graph G_d. The graph G_d consists of the isomorphic (self-similar) set of graphs g_0 and is called fractal.

The elastic systems are transformed in time, both in the direction of increasing capacity with an increasing workload (the system uses reserved resources and is deployed) and in the direction of decreasing capacity (the system frees up resources and folds).

Let the fractal dynamic graph-based model $G_d = f_d(g_0)$ be given and the iterated function f simulate the increasing capacity. Then decreasing capacity is simulated by the inverse function $f_d^{-1}(g_0)$. This forms an elastic model.

Def 4. The elastic fractal graph-based model G_d^e is as follows:

$$G_d^e = \begin{cases} f_d(g_0), if\ workload\ increasing \\ f_d^{-1}(g_0), if\ workload\ decreasing \end{cases} \quad (2)$$

In general, this is a dynamical system that is based on dynamics of fractal graphs.

B. Tensor presentation of elastic fractal graphs

Additional information about the relationships between vertices and their hierarchical structure, which is not present in the structural representation, is provided by the tensor representation of fractal graphs, see Table 1 column Graph. Higher-order interactions between vertices, that cannot be represented by traditional graph methods, are captured by the tensor-based approach, providing a more complete understanding of the system's behavior.

The simplest way to represent a graph is with an adjacency matrix. It describes which vertices are adjacent to which other vertices in a graph and consists of n×n real numbers arranged in n rows and n columns, where n×n are called its dimensions. The places in the matrix where the numbers are is called entries. The entry in row i and column j of the matrix

A is denoted by a_{ij}. An n×1 matrix is called a column vector of order n; similarly, a 1×n matrix is a row vector of order n.

Def 5. ***A*** (G) = [a_{ij}] is an adjacency matrix of the graph G. The rows and the columns of ***A***(G) are indexed by V in the following way:

$$A(G) = [a_{ij}] = \begin{cases} 1, if\ (v_i, v_j) \in E \\ 0, if\ (v_i, v_j) \notin E \end{cases} \quad (3)$$

If the a_{ij} entry of ***A***(G) is 0 then vertices i and j nonadjacent, and the a_{ij} entry is 1 for i and j adjacent.

If a_{ii} entry of ***A***(G) is 1 for i = j then the graph G is a loop-digraph, i.e., it is an undirected graph in which an edge can begin and end at the same vertex.

Graph G is a simple if it is an unweighted, undirected graph containing no graph loops or multiple edges. A simple graph may be either connected or disconnected.

If a_{ij} entry of ***A***(G) is 0 for i ≠ j then ***A***(G) is a diagonal matrix. The diagonal matrix is denoted

$$diag(a_{11}, a_{22}, \dots, a_{nn}) = \begin{bmatrix} a_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{nn} \end{bmatrix}$$

If a_{ij} entry is 1 for all i, this matrix reduces to the identity matrix $\boldsymbol{I}_n$ of the order n.

$$I_1 = [1], I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$I_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \dots, \quad \begin{bmatrix} a_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{nn} \end{bmatrix}$$

An adjacency matrix ***A***(G) for a simple graph and a loop-digraph is a logical matrix, that is, one whose elements are all either 0 or 1. The adjacency matrix ***A***(G), for an undirected graph G is symmetrical about the main diagonal. This is because if vertex i is adjacent to vertex j, then j is adjacent to i.

A multidimensional matrix is a tensor related to a vector space. The order of a tensor, given by m, is defined by the number of directions (dimensions) it has. Each dimension is called a mode. The adjacency matrix has 2 dimensions. The tensor model occurs because the sequence d = 1,2,...,n of the G_d^e presents a 1×d row vector.

Def. 6. Tensor model ***T*** (G_d^e) of elastic fractal graphs G_d^e combines the adjacency matrix with 1×d row vector and it is a 3-order tensor.

2. Tensor Models Based on Elastic Fractal Graphs

In this section, both tensor models of elastic fractal graphs based on the Kronecker product and algebraic operations are considered. The properties of adjacency matrices associated with the generation of graphs are revealed. Their comparative analysis is given.

A. Tensor models based on Kronecker elastic fractal graphs

The Kronecker product of two graphs is the product of its respective adjacency matrices.

Def. 7. Kronecker product of an adjacency matrices ***A*** = [$a_{i,j}$] and ***B*** of sizes $n \times n$ and m×m respectively is an adjacency matrix ***C*** of dimensions (n ▪ m)×(n ▪ m) is given by

$$C = A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nm}B \end{bmatrix}$$

Kronecker product is a way of generating self-similar matrices. Let initial graph g_0 is a pattern, which defines by adjacency matrix $\boldsymbol{K}_0$ (g_0), see Fig. 1. The graph g_0 is a loop-digraph: each vertex has loop edge.

Fig. 2 shows intermediate stage of Kronecker product of initial graph g_0 with itself $g_1 = g_0 \otimes g_0$. Each of the v_1, v_2, v_3 vertex gets expanded into 3 subgraphs similar to g_0, which are then linked.

Fig. 3 shows the corresponding graph $g_1 = g_0 \otimes g_0$ and adjacency matrices $\boldsymbol{K}_1(g_1) = \boldsymbol{K}_0(g_0) \otimes \boldsymbol{K}_0(g_0)$

$$K_0(g_0) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

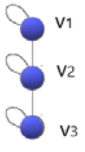


Fig. 1. Adjacency matrix $K_0(g_0)$ and graph g_0

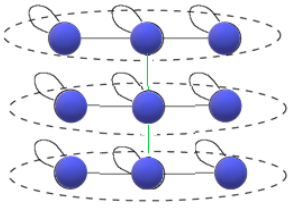
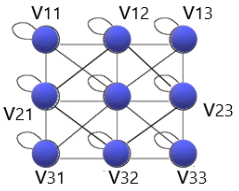


Fig. 2. Intermediate stage of Kronecker product



$$K_1 = K_0 \otimes K_0 = \begin{bmatrix} K_0 & K_0 & 0 \\ K_0 & K_0 & K_0 \\ 0 & K_0 & K_0 \end{bmatrix}$$

Fig. 3. Kronecker product: $g_1 = g_0 \otimes g_0$ and $\mathbf{K}_1(g_1) = \mathbf{K}_0(g_0) \otimes \mathbf{K}_0(g_0)$

Consider a fractal graph model that is based on the Kronecker product. The left and right parameters of the Kronecker product for the graph g_1 are the same: $g_1 = g_0 \otimes g_0$. Because of it, the iterated Kronecker product $\otimes$ for graphs can be presented as $g_1 = \otimes(g_0)$. Then a dynamic fractal graph K_d is obtained as follows:

$$K_d = \otimes_d (g_0) \quad (4)$$

The inverse product $K_d = \otimes_d^{-1}(g_0)$, stated in terms of the product $\otimes_d^{-1}(g_0) = \otimes_{d-1}(g_0)$, if $i > 0$. As a result, the elastic Kronecker product-based graph K_d^e is as follows:

$$K_d^e = \begin{cases} \otimes_d (g_0), if\ workload\ increasing \\ \otimes_d^{-1} (g_0), if\ workload\ decreasing \end{cases} \quad (5)$$

Fig. 4. shows the tensor $\boldsymbol{T}$ (K_d^e (g_0)) of elastic fractal graph K_d^e, where g_0 is defined as in Fig. 1.

The Kronecker product has the following fractal graph properties:

- If a zero-entry is determined in an adjacency matrix $\boldsymbol{K}(g_0)$, then a product by zero gives 0. As a result, it is preferable that the initial graph contain as few zero-entry points as possible.
- Therefore, loop-digraphs are often used for modeling, as they contain fewer zero-entry compared to simple graphs.
- The growth number of the vertices (n,n) depends significantly on the order of the initial subgraph g_0.
- Mapping to metric space gives similarity in accordance with fractal dimension.

B. Tensor models based on algebraic operations over elastic fractal graphs

Operations over a graph G and how they are translated into tensor operations are defined. The operations are systematized into algebra GA.

Def. 8. Let GA be an algebra over a graph G, with the set of operations Δ:

$$GA = < G, \Delta >, \text{ where} \quad (6)$$

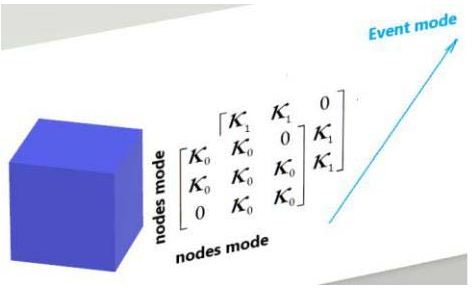


Fig. 4. Tensor $\boldsymbol{T}(K_d(g_0))$, d = 2

$\Delta = \{ \equiv, \leftrightarrow, \div \}$ is a set of uniquely invertible operations. These operations are described below.

$\equiv$ is a replication ($\equiv^{-1}$ is an invertible replication). This is a simple unary operation on the graph g. g' = g$\equiv$, where component g' is a replication of the graph g. If its writing was omitted by just writing down the component g' the operation is executed by default.

Let $\boldsymbol{A}$ (g) be an adjacency matrix of the subgraph g ∈ G. The replication $\boldsymbol{A}$ (g)$\equiv$ is a block diagonal matrix whose off-diagonal blocks are zero matrices and diagonal blocks are equals to $\boldsymbol{A}$ (g') = $\boldsymbol{A}$ (g). The diagonal matrix is denoted:

$$A(G)^c = \begin{bmatrix} A(g) & 0 \\ 0 & A(g') \end{bmatrix} = diag(A(g)_{11}, A(g')_{22})$$

The upper index ^{c} denotes that matrix $\boldsymbol{A}$ (G)^c has only two blocks, and graph G has only two equal components, g and g'.

$\leftrightarrow$ is a connection of the two components by vertices. ($\leftrightarrow^{-1}$ is an invertible connection or disconnection). This is a simple binary operation between the vertices of two components, g $\leftrightarrow$ g'.

A square zero block matrix $\boldsymbol{I}$ is formed for the connection of two components, g and g'. The size of $\boldsymbol{I}$ is the same as that of $\boldsymbol{A}$ (g). The connected vertices of the components g and g' are correspondent by the entries of the matrix $\boldsymbol{I}$ (g $\leftrightarrow$ g') that is assigned 1. The matrix $\boldsymbol{I}$ (g $\leftrightarrow$ g') is entered in a descending diagonal from right to left instead of 0.

$$A(G) = \begin{bmatrix} A(g) & I(g \leftrightarrow g') \\ I(g \leftrightarrow g') & A(g') \end{bmatrix}$$

$\div$ is a subdivision of the edge by inserting a new vertex into the edge ($\div^{-1}$ is an invertible subdivision). The vertex does not belong to the graph. It is a composition operation of the two components with a new vertex.

The two column vectors $\boldsymbol{I}$ the same order are formed for the composition of two components, g and g'. If $\boldsymbol{A}$ (g) has an order d, then the order of the column vector $\boldsymbol{I}$ is d, if d = 0 then $\boldsymbol{I}$ = [0].

The column vector $\boldsymbol{I}$ has an entry equal to 1 iff its vertex is connected to $\boldsymbol{A}$ (g) with $\boldsymbol{A}$ (g'). $\boldsymbol{I}^{\mathrm{T}}$ is the transpose of the column vector $\boldsymbol{I}$ into a row vector. It should be noted that for the composition of the two components, one vertex is enough, i.e., only two entries equal to 1.

$$A(G) = \begin{bmatrix} A(g) & 0 \\ 0 & A(g') \end{bmatrix} \div [\mathrm{I}] = \begin{bmatrix} A(g) & 0 & I \\ 0 & A(g') & I \\ I^T & I^T & 0 \end{bmatrix}$$

The result of the operation subdivision of the edge is an adjacency matrix $\boldsymbol{A}$ (G) of tree-like graph G. The row vector $\boldsymbol{I}^{\mathrm{T}}$ gives a symmetrical adjacency matrix.

$G_d = \Delta_d(g_0)$ is a fractal graph, as shown in formula 1.

Consider fractal graphs and the algebraic operations that are used to synthesized them from the initial graph g_0 with one vertex.

Fundamental fractal graph algorithms such as linear graphs, grids, hypercubes, and trees are written down in an ordered sequence by the algebraic operations Δ.

The linear, grid fractal graphs are represented in the cube topology.

Algorithm 1. The linear fractal graph $L_d = \Delta_d(g_0)$, where $\Delta = (v \in g_0) \leftrightarrow (v' \in g_0')$, see Fig. 5. L_d is simple graph, d = 3.

An adjacency matrix $\boldsymbol{L}(L_d)$ using block matrices is written as follows::

$$L_0(g_0) = [0],\ L_1(g_0) = \begin{bmatrix} L_0 & I_1 \\ I_1 & L_0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

$$L_2(g_0) = \begin{bmatrix} L_1 & I_2^L \\ I_2^L & L_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \ldots,$$

$$L_d(g_0) = \begin{bmatrix} L_{d-1} & I_d^L \\ I_d^L & L_{d-1} \end{bmatrix}$$

A square block matrix $\boldsymbol{I}_1^{\mathrm{L}} = 0$, and $\boldsymbol{I}_d^{\mathrm{L}}$ for d > 1 has d,d-entry equal to 1 with all other entries equal to 0.

Algorithm 2. The grid fractal graph $GR_d = \Delta_d(g_0)$, where $\Delta = (v_i \in g_0) \leftrightarrow (v_i' \in g_0')$, $(v_j \in g_0) \leftrightarrow (v_j' \in g_0')$. v_i, v_j, $i \neq j$ – two pairs of isomorphic vertices of the graph g_0 and component g_0', see Fig. 6. GR_d is simple graph, d = 3.

An adjacency matrix $\boldsymbol{G}(GR_d)$ using block matrices is written as follows:

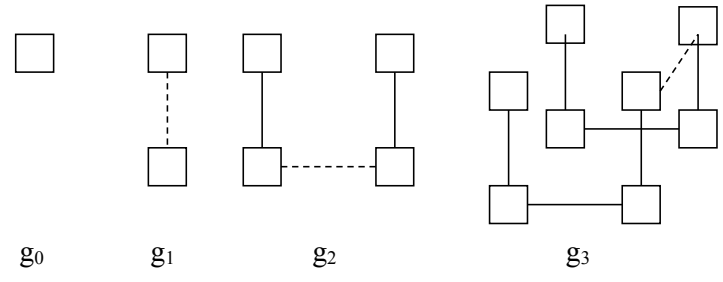


Fig. 5. The linear fractal graph $L_d = \Delta_d(g_0)$, d = 3

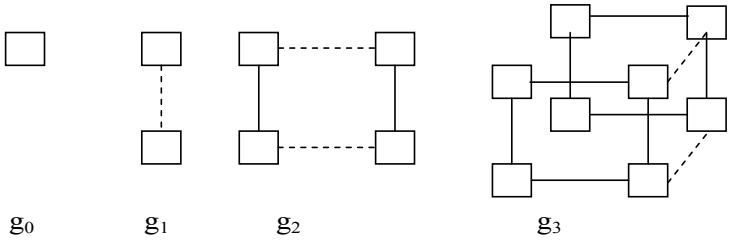


Fig. 6. The grid fractal graph $GR_d = \Delta_d(g_0)$, d= 3

$$G_0(g_0) = [0], G_1(g_0) = \begin{bmatrix} G_0 & I_1 \\ I_1 & G_0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

$$G_2(g_0) = \begin{bmatrix} G_1 & I_2 \\ I_2 & G_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \ldots,$$

$$G_d(g_0) = \begin{bmatrix} G_{d-1} & I_d^G \\ I_d^G & G_{d-1} \end{bmatrix}$$

A square block matrix $\boldsymbol{I}_d^G$ for d =3 and the grid width equals 2 has d-1,d+1-entry equal to 1, and d,d+2-entry equal to 1.

Algorithm 3**.** The hypercube fractal graph $H_d = \Delta_d(g_0)$, where $\Delta = \forall\ v,v'\ (v \in g_0) \leftrightarrow (v' \in g_0')$, see Fig. 7. H_d is simple graph, d = 3.

An adjacency matrix $\boldsymbol{H}(H_d)$ using block matrices is written as follows:

$$H_0(g_0) = [0], H_1(g_0) = \begin{bmatrix} H_0 & I_1 \\ I_1 & H_0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

$$H_2(g_0) = \begin{bmatrix} H_1 & I_2 \\ I_2 & H_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \ldots,$$

$$H_d(g_0) = \begin{bmatrix} H_{d-1} & I_d \\ I_d & H_{d-1} \end{bmatrix}$$

For the hypercube, a block matrix $\boldsymbol{I}_n$ is identity matrix, see Def. 5.

For the considered algorithms, operations Δ iterate the initial graph g_0, which has one vertex.

Algorithm 4**.** The binary tree fractal graph $B_d = \Delta_d(g_0)$, where $\Delta = (v \in g_0) \leftrightarrow (v' \in g_0'), (v \in g_0) \div (v' \in g_0')$. v, v' – highlighted not only the graph's vertices and the component g_0 and g_0' but also edge e = (v, v') for operation ÷. A comma separates the two operations on the connection of the components and the subdivision of the edge e = (v, v') of the graph B_d, see Fig. 8, d = 2.

The algorithm 4 is shown in two steps: first, the connection operation is performed, and then the subdivision is performed. The binary tree graph B_d is a simple graph.

In the Fig. 8, the edges vertices of subdivision operations are marked with a dotted line. An adjacency matrix $\boldsymbol{B}(B_d)$ using block matrices is written as follows::

$$B_0(g_0) = [0],$$

$$B_1(g_0) = \begin{bmatrix} B_0 & 0 \\ 0 & B_0 \end{bmatrix} \div [I_0^B] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \div [I_0^B] = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \ldots,$$

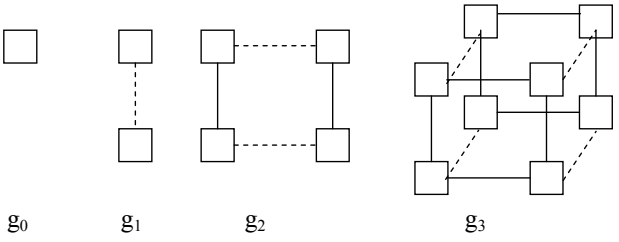


Fig. 7. The hypercube fractal graph $H_d = \Delta_d(g_0)$, d= 3

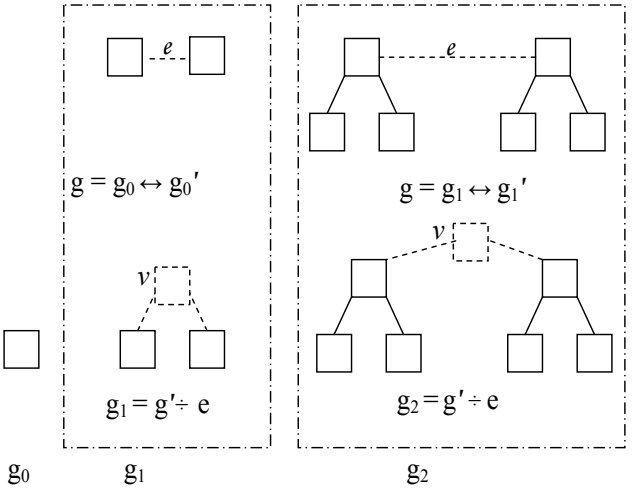


Fig. 8. The binary tree fractal graph B_d, d = 2

$$B_d(g_0) = \begin{bmatrix} B_{d-1} & 0 \\ 0 & B_{d-1} \end{bmatrix} \div [\mathrm{I}^{\mathrm{B}}_{\mathrm{d}-1}]$$

The subdivision ÷ operation of the edge for binary tree has next algorithm:

1. The two column vectors $\boldsymbol{I}^{\mathbf{B}}{}_{\mathrm{d}}$ the order d are formed.

2. If d = 0 then $\boldsymbol{I}^{\mathbf{B}}{}_{0}$ = [1], if d > 0, then $\boldsymbol{I}^{\mathbf{B}}{}_{\mathrm{d}}$ =[$\mathrm{i}_{1.\mathrm{d}\text{-}1}$] = 1, for each column vectors.

3. The subdivision operation is executed.

The elastic fractal graph-based model (Def. 4) and equivalent tensor models can be obtained using algebraic invertible operations. The Table 2 shows algebraic types of fractal graph-based models.

The algebraic fractal graphs have the following properties:

- The tensor representation is specified in the form of block-type tensors with changeable dimensions for the replication operation. As a result, the presence of zero-entry points in the initial graph is not so important as it is for the Kronecker product. For modeling, both simple graphs and loop-digraphs can be used.
- The growth number of the vertices of the graph after algebraic operations depends on the order of the initial graph g_0, and depends on the consequence of replication operations as well as subdivision operations.

3. Analysis of Fractal Graphs

Analysis of elastic fractal graphs must take into account the properties of graphs [18, 19], and tensor models [20].

A. Analysis properties of fractal graphs

Self-similarity is the most important property of fractal graphs. The properties that remain unchanged under the transformation of fractal graphs are the initial components of the graphs from which they were synthesized. Similarity gives the opportunity to reconstitute a damage model based on a graph. Elastic fractal graphs have the ability to increase and decrease in dimensions.

B. Tensor Model Analysis

In the paper, adjacency square matrices and block square matrices and their corresponding graphs were considered. Adjacency square matrices are symmetric, diagonal matrices, and block square matrices. The matrix fractality of graphs can be specified both by the Kronecker product and by block matrices.

Elastic tensor models have the ability to increase and decrease their dimensions without decomposition.

Conclusion

The contribution of this paper refers to the development of graph-dynamic models. In this regard, it is important to convert large-scale problems to simplified fractal models. This research yielded a generalization of many techniques to fractal graphs and their evolution. The following are ways in which the paper contributes:

- The extension of fractal algebra, using tensor algebra and the Kronecker product;
- Introducing various types of tensors and operations that facilitate the formation of fractal graphs;
- Identifying two types of fractal tensors: nested fractal tensors, which are based on Kronecker products, and symmetric fractal tensors, which are based on copying the original adjacency matrices. These new methods expand the possibilities of utilizing fractal graphs.

The author envisions continuing work in two directions: managing and planning elastic calculations based on fractal tensors, and identifying the properties of tensor operations on adjacency matrices of fractal graphs.

Table 2. Fractal type graph-based models

Graph type	Fractal graph network models		
	Fractal graph	***Elastic Fractal graph***	***Tensor***
Linear	Algorithm 1. $L_d = \Delta_d(g_0)$	L_d^e	$\boldsymbol{T}(L_d^e)$
Grid	Algorithm 2. $GR_d = \Delta_d(g_0)$	GR_d^e	$\boldsymbol{T}(GR_d^e)$
Hypercube	Algorithm 3. $H_d = \Delta_d(g_0)$	H_d^e	$\boldsymbol{T}(H_d^e)$
Tree	Algorithm 4. $B_d = \Delta_d(g_0)$	B_d^e	$\boldsymbol{T}(B_d^e)$

References

1. Becker S., Brataas G., and Lehrig S. (Eds.). Engineering Scalable, Elastic, and Cost-Efficient Cloud Computing Applications. The CloudScale Method, Springer International Publishing AG 2017, 187p.
2. Semenov A.S. 2017. Fractal Petri Nets [4th Int. Conf. Control, Decision and Information Technologies. Proceedings] Barcelona, Spain, 2017, pp. 1174 – 1179.
3. Semenov A.S. 2020. Graph-based Dynamic Analysis of Elastic Systems. [7th Int. Conf. Control, Decision and Information Technologies. vol 1 Conf. Paper] Publisher: IEEE pp. 65-70, doi: 10.1109/CoDIT49905.2020.9263986.
4. Semenov A.S. Fractal Analysis and Programming of Elastic Systems Using Container-Component Model. 2021. [Smart Innovation, Systems and Technologies, Jain, L. C., Margarita N. Favorskaya, Ilia S. Nikitin, and Dmitry L. Reviznikov, Eds. Springer Nature EBook], 2021, pp. 307-320. https://doi.org/10.1007/978-981-33-4826-4.
5. Laub, A. J. Matrix Analysis for Scientists and Engineers. Calofornia, Devis SIAM, 2005, pp. 139-144.
6. Liu Y., Liu J., Long Z., Zhu C., Tensor Computation for Data Analysis. Springer Nature Switzerland AG 2022.
7. Leskovec J., Chakrabarti D., Kleinberg J., Faloutsos C., Ghahramani Z., "Kronecker Graphs: An Approach to Modeling. [Journal of Machine Learning Research] 11, 2010. pp. 985-1042.
8. Semenov A.S. Prototype based Programming with Fractal Algebra, [AIP Conf. Proc.] vol.. 2181-1, 2019. https://doi.org/10.1063/1.5135669.
9. Semenov A,S. Essentials of Fractal programming, [Smart Innovation, Systems and Technologies] L. C. Jain et al. (eds.), Advances in Theory and Practice of Computational Mechanics, Springer-Verlag, 173, 2020, pp. 373-386. https://doi.org/10.1007/978-981-15-2600-8_25.
10. R.B. Bapat, Graphs and Matrices. New York: Springer, 2010.
11. Mortveit, H., and C. Reidys. An Introduction to Sequential Dynamical Systems. NY, NY: Springer. 2007.
12. Kuhlman, K., Mortveit, H., et al. A general-purpose graph dynamical system modeling framework,[Proc. of the 2011 Winter Simulation Conf]
13. Papineni, K., Worah, P. A Dynamical System on Bipartite Graphs, [Proc. of the 27th ACM Int. Conf. on Information and Knowledge Management], 2018, Torino, Italy. pp. 1479-1482.
14. Sizemore, A.E., Bassett, D.S. Dynamic graph metrics: Tutorial, toolbox, and tale. [Journal of the NeuroImage. Elsevier] vol. 180, 2018. pp. 417–427. www.elsevier.com/locate/neuroimage
15. Pignolet, Y.,A., et al. The Many Faces of Graph Dynamics. [Social and Information Networks] 2017. arXiv:1506.01565
16. Despreaux, S., Maignan. A. Dynamical Systems Based on Dynamic Graphs. 2009. https://www.researchgate.net/publication/254740120
17. Leskovec, J., Faloutsos, C. Scalable Modeling of Real Graphs using Kronecker Multiplication, [Proc. of the 24 th Int. Conf. on Machine Learning] Corvallis, OR, 2007.
18. Charu Aggarwal and Karthik Subbian. Evolutionary network analysis: A survey, [ACM Comput. Surv.] vol 47,1. 2014. DOI: http://dx.doi.org/10.1145/2601412
19. Heckel, R., Taentzer, G. (Eds.). Graph Transformation, Specifications, and Nets. Springer International Publishing AG, part of Springer Nature. 2018. 331p.
20. Fridtjov I. Tensor Analysis. Springer Nature Switzerland AG 2019

Semenov Alexander S. is with the Computational Mathematics and Programming Department, Moscow Aviation Institute (National Research University), MAI, Moscow, 125993 Russia, candidate of Physical and Mathematical Sciences, Associate Professor. Research interests: development of reconfigurable systems, elastic and quantum computing using fractal algebra (fractoids). E-mail: Semenov_Alex@yahoo.com

Тензорные модели фрактальных графов для эластичных сетей

А. С. Семенов

Московский авиационный институт (Национальный исследовательский университет), Москва, Россия

Аннотация. В статье представлены результаты исследований фрактальных (самоподобных) графов применительно к эластичным вычислениям. Характерной особенностью таких графов, является возможность их разворачивания (увеличения размерности) и сворачивания (уменьшения размерности). Рассмотрены два подхода к формированию фрактальных графов: на основе произведения Кронекера и фрактальной алгебры. Представлена взаимосвязь алгебраических операций формирования фрактальных графов (линейные графы, сетки, гиперкубы и деревья) с тензорными операциями и тензорным представлением на основе интеграции матриц смежности и вектора событий эластичной системы. Введены определения соответствующих типов динамически изменяемых тензоров. Проведен анализ свойств эластичных фрактальных графов и связанных с ними тензорных моделей.

Ключевые слова: графы Кронекера, фрактальные графы, фрактальная алгебра, эластичные сети, тензорные модели.

DOI 10.14357/20718632230412 **EDN** SSMYLI

Литература

1. Беккер С., Братаас Г., Лерриг С. (ред.). Создание масштабируемых, эластичных и экономически эффективных приложений для облачных вычислений. Метод CloudScale, Springer International Publishing AG 2017, 187с.
2. Семенов А.С. 2017. Фрактальные сети Петри [4-я международная конференция по управлению, принятию решений и информационным технологиям. Сборник трудов]. Барселона, Испания, 2017, С.1174-1179.
3. Семенов А.С. 2020. Графо-ориентированный динамический анализ эластичных систем. [7-я международная конференция по управлению, принятию решений и информационным технологиям, том 1 Конференция докладов] Издательство: IEEE С. 65-70, doi: 10.1109/CoDIT49905.2020.9263986.
4. Семенов А.С. Фрактальный анализ и программирование эластичных систем с использованием контейнерно-компонентной модели. 2021. [Smart Innovation, Systems and Technologies, Ред. Джайн, Л. С., Маргарита Н. Фаворская, Илья С. Никитин и Дмитрий Л. Ревизников. Springer Nature EBook], 2021, с. 307-320. https://doi.org/10.1007/978-981-33-4826-4.
5. Лоб, А.Дж. Матричный анализ для ученых и инженеров. Калифорния, Дэвис SIAM, 2005, С.139-144.
6. Лю Янь, Лю Цзянь, Лонг Чжу, Чжу Сы., Тензорные вычисления для анализа данных. Springer Nature Switzerland AG 2022.
7. Лесковец Дж., Чакрабарти Д., Клейнберг Дж., Фалоутсос К., Гахрамани З., "Графы Кронекера: Подход к моделированию. [Журнал исследований машинного обучения] 11, 2010. с. 985-1042.
8. Семенов А.С. Прототипно-ориентированное программирование с фрактальной алгеброй, [AIP Conf. Proc.] том 2181-1, 2019. https://doi.org/10.1063/1.5135669.
9. Семенов А. Основы фрактального программирования // Умные инновации, системы и технологии: Перспективные направления развития вычислительной механики: теория и практика. Л. Джайн и др. (ред.), Springer-Verlag, 2020, том 173, С. 373-386. https://doi.org/10.1007/978-981-15-2600-8_25.
10. Р.Б. Бапат, Графы и Матрицы. Нью-Йорк: Springer, 2010.
11. Мортвейт Х., и К. Рейдис. Введение в последовательные динамические системы. Нью-Йорк, Нью-Йорк: Springer. 2007.
12. 13. Кульман, К., Мортвейт, Х. и др. Фреймворк моделирования графовых динамических систем общего назначения [Доклады зимней конференции по моделированию систем 2011 года].
13. Папинени, К., Ворах, П. Динамическая система на двудольных графах, [Материалы 27-й международной конференции ACM по информационным технологиям и управлению знаниями], 2018, Турин, Италия. С. 1479-1482.
14. Sizemore, A.E., Bassett, D.S. Динамические метрики графов: учебник, инструментарий и рассказ. [Журнал NeuroImage. Elsevier] т. 180, 2018. С. 417–427. www.elsevier.com/locate/neuroimage.
15. Пиньоле Й.А. и др. Многие лица динамики графов. [Социальные и информационные сети] 2017. arXiv:1506.01565.
16. Деспро, С., Меньян, А. Динамические системы на основе динамических графов. 2009. https://www.researchgate.net/publication/254740120.
17. Лесковец, Дж., Фалоутсос, К. Масштабируемое моделирование реальных графов с использованием умножения Кронекера, [Проц. 24-й международной конференции по машинному обучению] Корваллис, Орегон, 2007.
18. Чару Аггарвал и Картик Суббиан. Эволюционный анализ сетей: обзор, [ACM Comput. Surv.] том 47,1. 2014. DOI: http://dx.doi.org/10.1145/2601412.
19. Хекель, Р., Тэнтзер, Г. (ред.). Трансформация графов, спецификации и сети. Издательство Springer International Publishing AG, часть Springer Nature. 2018. 331 с.
20. Фридтьев И. Тензорный анализ. Springer Nature Switzerland AG 2019.

Семенов Александр Сергеевич Московский авиационный институт (национальный исследовательский университет), МАИ, Москва, Россия. Доцент, кандидат физико-математических наук. Область научных интересов: разработка реконфигурируемых систем, эластичных и квантовых вычислений с использованием фрактальной алгебры (фрактоидов). E-mail: Semenov_Alex@yahoo.com