computers

MDPI

Article

Fractal Aspects in Classical Parallel Computing

Mitică Craus [1,*], Vlad-Sergiu Bîrlescu [1] and Maricel Agop [2]

[1] Department of Computer Science and Engineering, "Gheorghe Asachi" Technical University of Iaşi, 27 Dimitrie Mangeron Street, 700050 Iasi, Romania; bvladsergiu@yahoo.com

[2] Department of Physics, "Gheorghe Asachi" Technical University of Iaşi, 67 Dimitrie Mangeron Street, 700050 Iasi, Romania; m.agop@yahoo.com

[*] Correspondence: craus@cs.tuiasi.ro; Tel.: +40-232-232-430

Academic Editor: Yevgeniya Kovalchuk
Received: 20 April 2016; Accepted: 7 September 2016; Published: 12 September 2016

Abstract: In this paper, we prove that many parallel communication topologies and several parallel algorithms have fractal properties, which leads to the idea that there is a strong connection between the classical parallel computing and a plausible computing on a fractal medium. Furthermore, we show that some parallel algorithms are suitable for incursive and hyperincursive computation, which links parallel computing to quantum computing and a possible fractal computing.

Keywords: fractals; fractal systems; parallel networks; parallel algorithms; incursive computation; quantum computing

MSC: 68Q10; 68Q10; 68Q99

1. Introduction

The pressure of complex problems, derived from technological progress and globalization, forced the Information and Communication Technology (ICT) world to build adequate hardware and software support. Parallel computing is one of the pillars of the High Performance Computing (HPC) concept. Software development for multicomputer and multiprocessor systems is not easy and requires an adaptation effort from the specialists. Parallel programming is fundamentally different from sequential programming. The programmers must increase their efforts so that the specific technologies regarding parallel computing systems become more accessible to the ones that need this potentially unlimited computing power. Nowadays, multiprocessor systems, multicore processors and graphic cards are developing at a thundering rhythm. They tend to dominate the HPC world. GPGPU computing (General Purpose Graphic Processing Unit Computing) prefigures itself as a new challenge of the present. Multiprocessor systems integrated on a single chip appear on the horizon. However, the perspective of the development of other new technologies is very promising. Non-deterministic computing is widely studied, but very little exploited. Quantum computing tries to fill this gap.

As is well known, the quantum medium is the base of the theoretical fundamentation of the quantum computer. In a recent paper [1], we have defined the fractal medium, and we have proven that it has multiple computational properties: bistability, self-reproduction, capacity to memorize, self-similarity, polarization, etc. Thus, the fractal medium could be the host for developing universal computers to solve the inconveniences of classical and quantum computers.

Any structural unit of a complex system is in a permanent interaction with the "sub-fractal level" (fractal medium) through the specific fractal potential. For motions on Peano curves at the Compton scale, the fractal medium is assimilated to the sub-quantum level [2]. The fractal medium is identified with a non-relativistic fractal fluid described by the fractal momentum and the fractal states' density conservation laws. In the fractal hydrodynamics approach of the scale relativity theory with an arbitrary constant fractal dimension, the fractal information, as a measure of the order degree of a complex system through its motion curves' fractality, initially unstructured and made not explicit on the fractal medium, becomes structured and so made explicit on the fractal medium in the form of fractal patterns, by means of fractal spontaneous symmetry breaking. Moreover, the structured relations among the fractal medium patterns induce a special fractal topology, which generates the fundamental fractal logic elements (fractal bit, fractal gates, etc.). In such a perspective, the quantum logic becomes a particular case of the fractal logic, for a given resolution scale dt/τ.

A fractal bit is a non-differentiable two-level system that, in addition to two pairwise orthogonal fractal states $|0(dt/\tau)\rangle$ and $|1(dt/\tau)\rangle$ in the Hilbert fractal space $C^2(dt/\tau)$, can be set to any superposition of the form:

$$|\Psi(\frac{dt}{\tau})\rangle = a(\frac{dt}{\tau})|0(\frac{dt}{\tau})\rangle + b(\frac{dt}{\tau})|1(\frac{dt}{\tau})\rangle,\ a(\frac{dt}{\tau}), b(\frac{dt}{\tau}) \in C(\frac{dt}{\tau}) \tag{1}$$

where:

$$\begin{aligned} \langle 0(\frac{dt}{\tau})|0(\frac{dt}{\tau})\rangle = 1, \langle 1(\frac{dt}{\tau})|1(\frac{dt}{\tau})\rangle = 1 \\ \langle 0(\frac{dt}{\tau})|1(\frac{dt}{\tau})\rangle = 0, \langle 1(\frac{dt}{\tau})|0(\frac{dt}{\tau})\rangle = 0 \end{aligned} \tag{2}$$

Since $|\Psi(\frac{dt}{\tau})\rangle$ is normalized,

$$\langle \Psi(\frac{dt}{\tau})|\Psi(\frac{dt}{\tau})\rangle = 1 \tag{3}$$

we have

$$|a(\frac{dt}{\tau})|^2 + |b(\frac{dt}{\tau})|^2 = 1 \tag{4}$$

Any non-differentiable two-level system is a potential candidate for a fractal bit. The Boolean fractal states, zero and one, can be represented by a fixed fractal pair of orthogonal fractal states of the fractal bit. In the following, we get:

$$|0(\frac{dt}{\tau})\rangle = \begin{pmatrix} 1(\frac{dt}{\tau}) \\ 0(\frac{dt}{\tau}) \end{pmatrix}, |1(\frac{dt}{\tau})\rangle = \begin{pmatrix} 0(\frac{dt}{\tau}) \\ 1(\frac{dt}{\tau}) \end{pmatrix} \tag{5}$$

which is the standard fractal basis in $C^2(\frac{dt}{\tau})$. A parameter representation of a normalized fractal state $|\Psi(\frac{dt}{\tau})\rangle$ in $C^2(\frac{dt}{\tau})$ is given by the relation:

$$\begin{aligned} |\Psi(dt)\rangle &= \begin{pmatrix} e^{i\phi(\frac{dt}{\tau})} sin\Theta(\frac{dt}{\tau}) \\ cos\Theta(\frac{dt}{\tau}) \end{pmatrix} \\ &= e^{i\phi(\frac{dt}{\tau})} sin\Theta(\frac{dt}{\tau}) \begin{pmatrix} 1(\frac{dt}{\tau}) \\ 0(\frac{dt}{\tau}) \end{pmatrix} + cos\Theta(\frac{dt}{\tau}) \begin{pmatrix} 0(\frac{dt}{\tau}) \\ 1(\frac{dt}{\tau}) \end{pmatrix} \end{aligned} \tag{6}$$

The computational properties of a fractal medium are very promising [1]. The property of bistability is vital for a computational medium. The digital representation of the information is possible only if the host environment is able to have several stable states. The self-reproducing property makes possible the copy function in the computational medium. Furthermore, the capacity to reproduce itself is very important for distributed computing. It assures the possibility to communicate data. Moreover, the capacity to memorize is vital for a computational medium. Information storage is possible only if the host environment is able to have a hysteresis type property [3]. The self-similarity property is important for recursive and incursive computation. It is well known that it is difficult to implement recursion on parallel and quantum computers. Fractal computers could overcome this inconvenience due to their recursive nature. Polarization is an important property that allows one to modify the state of a computing medium to a desired one. Having stable states through bistability makes it possible to force the fractal medium towards such a state. This is useful in the input or write operations.

A fractal computer can be considered as a complex system hosted by a fractal medium, which processes fractal bits, using the fractal logic theory. Such a computer is suitable to infinite contraction and infinite extension because of the resolution scale included in its definition. The recursive and non-deterministic computation seems to be natural for fractal computers [4].

Furthermore, the fractal properties of several parallel communication topologies induce the idea that fractal computing would include classical parallel computing. Many parallel algorithms are suitable for incursive and hyperincursive computation, which creates connections between the classical parallel computing to quantum computing and fractal computing. In this paper, we will analyze such parallel communication topologies and parallel algorithms, their fractal and incursive features.

2. Self-Similarity Property of Fractals

The name fractal was given by Benoit Mandelbrot [5], and it is inspired from the Latin word "fractus" (broken, interrupted). Examples of fractals occurred long time before that date. They were regarded as a "strange mathematical construction". Benoit Mandelbrot's great merit was to discover their importance and the perspective of their applications [6].

Most fractals have the property of self-similarity. This concept was introduced by Gottfried Leibniz. Many fractals have a non-integer dimension. The first definition of such a notion was given by Hausdorff.

Let us remind about the building process of the Cantor set, M_c, in order to illustrate the properties we want to highlight. In the following, we use the method from [6].

Georg Cantor showed that there are sets of the power continuum, e.g., the M_c set that we shall build. Such a set can be put in one-to-one correspondence with a set R of points from an interval, let us say [0,1], which is discontinuous everywhere. R is called a rare set, i.e., any open interval I_0 that contains a point of R and has points that do not belong to R.

In order to construct M_c, we start with the closed interval [0,1] called the "initiator" and denoted by I_1 (Figure 1).

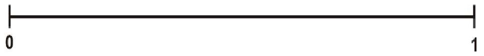


Figure 1. The initiator I_1 of Cantor's set.

The open interval (1/3, 2/3) is now cut out, and the set I_2 results (Figure 2).

Figure 2. Second stage of Cantor's set generation: set I_2.

Afterwards, we remove the open intervals (1/9, 2/9) and (7/9, 8/9) from the middle of the remaining sets and obtain the set I_3 (Figure 3).

0
1/9
2/9
1/3
2/3
7/9
8/9
1

Figure 3. Third stage of Cantor's set generation: set I_3.

Using the same method, we obtain the $I_4, I_5 \ldots$ sets. If the procedure indefinitely continues, M_c is the set of common points of these intervals.

$$M_c = \bigcap_{n=1}^{\infty} M_c^n \tag{7}$$

$$M_c^n = I_1 \cap I_2 \cap I_3 \cap \ldots \cap I_n \equiv \bigcap_{i=1}^{n} I_i \tag{8}$$

M_c is a self-similar set. Any closed interval that remained after we had eliminated the open intervals, after an infinite number of steps of the construction procedure, has a structure that is equivalent to M_c. Indeed, considering the one-to-one application of M_c with one of its subsets, which is Cantor's set built on the initiator $|p/3^n, (p+1)/3^n|$ with p conveniently chosen, we can easily find their equivalence (Figure 4).

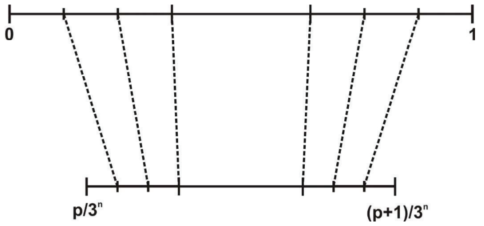


Figure 4. Cantor's set has the self-similarity property.

In order to define self-similarity, we make some comments. First, we have to limit to a space in which it is possible to define the likeness. Since the example we had considered is a set of the Euclidean space, this condition is satisfied. Second, we have to highlight that not any subset of the given set is alike to the whole set. For example, the set $\{1/3, 2/3, 7/9\}$ is a Cantor dust subset, but it does not resemble it.

Definition 1. *A fractal F, that is a subset of a metric space in which similarity is defined, is self-similar, if any open O for which $O \cap F \neq \emptyset$ contains a subset that is alike as F.*

A classic example is the fern. If we take a leaf part of it, it will have the same form of ribs as the whole leaf. However, this division is limited; it can not descend up to the size of a cell. In fact, the fern is a pre-fractal, and self-similarity is limited.

3. Self-Similarity in Parallel Communication Topologies

In order to prove self-similarity in parallel communication topologies, we will analyze those most used: full and complete binary tree, fat tree, mesh, hypercube, butterfly and banyan.

3.1. Full and Complete Binary Tree

This is one of the most used communication topologies in parallel computing.

Definition 2. *A full and complete binary tree is either a tree with three vertices where one of them is named the root, and it is linked by edges with the other two vertices, or it is a tree with three types of vertices: root, internal vertices and leaves. The root is linked by edges with two vertices that are the roots of two full and complete binary trees with the same number of vertices. Every internal vertex is linked by edges with three vertices: one of them is named the parent, and the others are the roots of two full and complete binary trees with the same number of vertices. Every leaf is linked by edges with one vertex named the parent.*

This definition is based on a decomposition principle and highlights the fractal character of the full and complete binary tree. For each node, the left and right subtrees are in their turn full and complete binary trees. The process of division in the tree cannot continue indefinitely as in the case of the Cantor set, but this disadvantage can be overcome by extending the tree with identical full and complete binary trees attached in pairs to each leaf.

Another definition is the following:

Definition 3. *A full and complete binary tree is either a tree with three vertices where one of them is named the root, and it is linked by edges with the other two vertices; or it is a tree formed by taking two full and complete binary trees with the same number of vertices (two identical full and complete binary trees), adding a vertex and than linking by edges the roots of the given binary trees.*

This definition is based on a composition principle. In such a way, the composing process can continue indefinitely. Every subtree is a full and complete binary tree. This gives to the full and complete binary tree a self-similar characteristic (Figure 5).

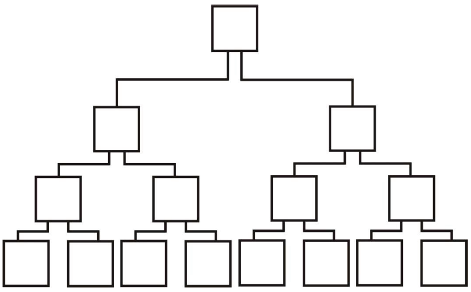


Figure 5. Full and complete binary Tree.

3.2. Fat Tree

Definition 4. *A fat tree is a full and complete binary tree where the vertices are linked by a set of edges. The cardinal of the set of edges that link a vertex to its parent is half of the cardinal of the set of edges that link the parent to its parent, in its turn.*

For each node, the left and right subtrees are in their turn fat trees. This confers a fractal character to the fat tree. The process of division in the tree cannot continue indefinitely. The presented extension in the case of the full and complete binary tree does not work.

Analogous to the full and complete binary tree, another definition can be given as follows:

Definition 5. *A fat tree is either a full and complete binary tree with three vertices or a graph formed by taking two fat trees with the same number of vertices (two identical fat trees), adding a vertex and then linking through a set of edges with the roots of the given fat trees. The cardinal of every set of edges is twice that of the cardinal of the set of edges that link the roots of the given fat trees with the roots of their subtrees.*

Similarly to a full and complete binary tree, the composing process can continue indefinitely. Every subtree is a fat tree. This gives to the fat tree a self-similar characteristic (Figure 6).

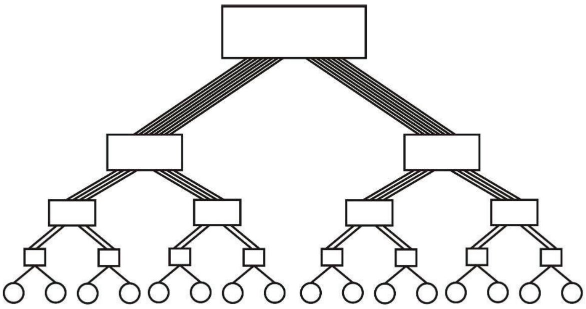


Figure 6. Fat tree.

3.3. Mesh

Definition 6. *A mesh is a communication topology where the set of vertices V is organized as a k-dimensional array through an indexing function* $I : V \to L = \{\langle i_{k-1}, i_{k-2} \ldots, i_0\rangle / 0 \leq i_j < n_j, j = k-1, k-2 \ldots 0\}$, *where* n_j *is the size of the array j-dimension,* $\{0, 1, \ldots, n_j - 1\}$, *and* $n = n_{k-1} * n_{k-2} * \cdots * n_0$. *We denote by* $v_{i_{k-1}, i_{k-2} \ldots, i_0}$ *the vertex given by* $I^{-1}(\langle i_{k-1}, i_{k-2} \ldots i_0\rangle)$. *On the j-th dimension, the vertex* $v_{i_{k-1}, i_{k-2} \ldots i_j \ldots, i_0}$ *is adjacent to vertex* $v_{i_{k-1}, i_{k-2}, \ldots, i_j+1, \ldots, i_0}$, *where* $i_j + 1 \leq n_j$, *and to vertex* $v_{i_{k-1}, i_{k-2}, \ldots, i_j-1, \ldots, i_0}$, *where* $i_j - 1 \geq 0$.

It is easy to observe that if we consider a piece of the k-dimensional array, defined by subsets of indices for each j-dimension, $\{inf_j^p, inf_j^p + 1, \ldots, sup_j^p\}$, the corresponding structure is also a mesh (Figures 7 and 8). In such a piece, vertex $v_{i_{k-1}, i_{k-2} \ldots i_j \ldots, i_0}$ is adjacent to vertex $v_{i_{k-1}, i_{k-2}, \ldots, i_j+1, \ldots, i_0}$, where $inf_j^p < i_j + 1 \leq sup_j^p$. Vertex $v_{i_{k-1}, i_{k-2} \ldots i_j \ldots, i_0}$ is adjacent to vertex $v_{i_{k-1}, i_{k-2}, \ldots, i_j-1, \ldots, i_0}$, where $inf_j^p \leq i_j - 1 < sup_j^p$.

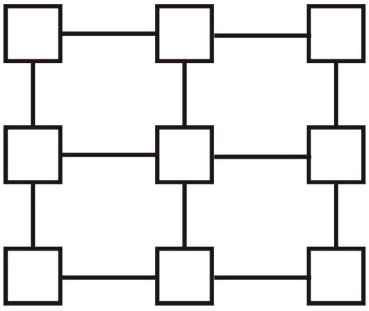


Figure 7. Two-dimensional mesh.

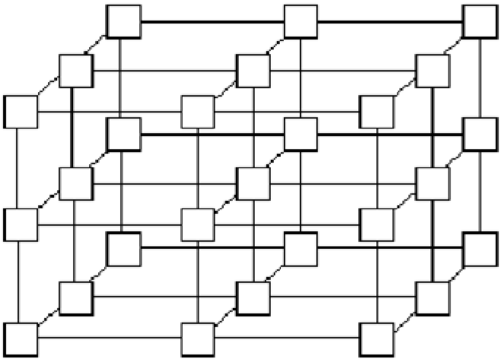


Figure 8. Three-dimensional mesh.

3.4. Hypercube

Definition 7. *A hypercube is a communication topology where the set of vertices V is of size* $n = 2^k$. *k is the dimension of the hypercube. Let us consider* $i_{k-1} i_{k-2} \ldots i_0$ *a binary representation of* $i \in \{0, \ldots, n-1\}$ *and* $i^{(j)}$ *the number whose binary representation is* $i_{k-1} i_{k-2} \ldots i_{j+1} \tilde{i}_j i_{j-1} \ldots i_0$, *where* $\tilde{i}_j = 1 - i_j, 0 \leq j < k$. *The edge set of the graph representing the hypercube is defined as follows: the vertex* v_i *is adjacent to vertices from the set* $\{v_{i^{(j)}}; 0 \leq j < k\}$.

A hypercube with k dimensions is composed of two $(k-1)$-dimensional hypercubes. The decomposing process can continue until $k = 0$.

Inversely, two hypercubes with $k - 1$ dimensions interconnected through corresponding vertices generate a k-dimensional hypercube. The composing process can continue indefinitely.

These properties give to the hypercube a self-similar characteristic (Figures 9–11).

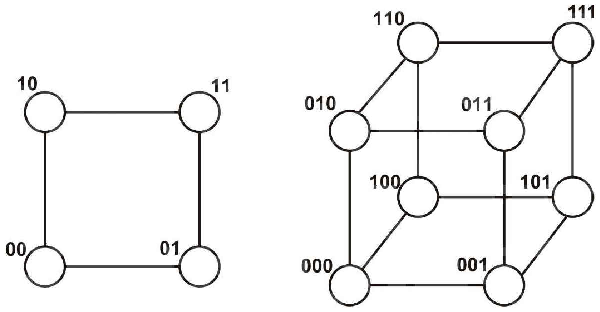


Figure 9. Two and three-dimensional hypercube.

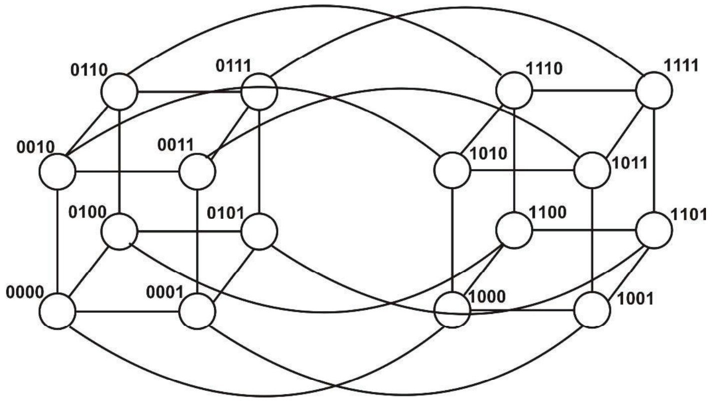


Figure 10. Four-dimensional hypercube.

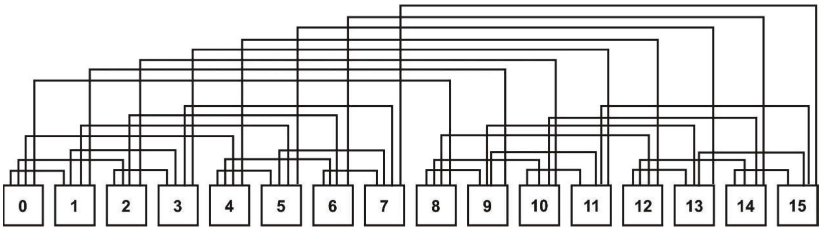


Figure 11. Four-dimensional hypercube linearized.

3.5. Butterfly

Definition 8. *The butterfly is a communication topology usually named a tree with multiple roots. The set of vertices, V, is organized as a two-dimensional array of size $n \times m$, where $n = 2^{m-1}$, via an indexation function $I : V \to L = \{(i,j) \ / \ 0 \leq i < n, 0 \leq j < m\}$. We denote by $v_{i,j}$ the vertex given by $I^{-1}((i,j))$. The vertex $v_{i,j}$, located on the column j, is connected to the column $j+1$ with the vertices $v_{i,j+1}$ and $v_{i \oplus 2^{m-j-2} + \lfloor i/2^{m-j-1} \rfloor x 2^{m-j-1}, j+1}$, where $\oplus$ means addition modulo 2^{m-j-1}.*

There is a single path between every pair of vertices situated on the left and right margin, respectively (Figure 12).

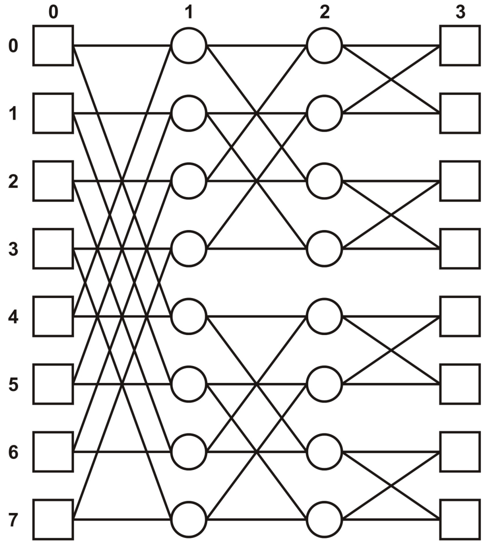


Figure 12. Butterfly.

An important remark is that the hypercube is a folding of the butterfly topology. We can obtain a hypercube from a butterfly by merging all butterfly vertices that are in the same row and then removing the edges that link the merged vertices.

3.6. Banyan

Definition 9. *Banyan is also a communication topology where the set of vertices, V, is organized analogously to butterfly as a two-dimensional array of size $n \times m$, where $n = 2^{m-1}$, via an indexation function $I : V \rightarrow L = \{(i, j) \ / \ 0 \leq i < n, 0 \leq j < m\}$. We denote by $v_{i,j}$ the vertex given by $I^{-1}((i, j))$. The vertex $v_{i,j}$, located on the column j, is connected to the column $j + 1$ with the vertices $V_{\lfloor i/2 \rfloor \oplus 0 + \lfloor i/2^{m-j-1} \rfloor x 2^{m-j-1}, j+1}$ and $v_{i \oplus 2^{m-j-2} + \lfloor i/2^{m-j-1} \rfloor x 2^{m-j-1}, j+1}$, where $\oplus$ means addition modulo 2^{m-j-1}.*

Similarly to butterfly, there is a single path between every pair of vertices situated on the left and right margin, respectively (Figure 13).

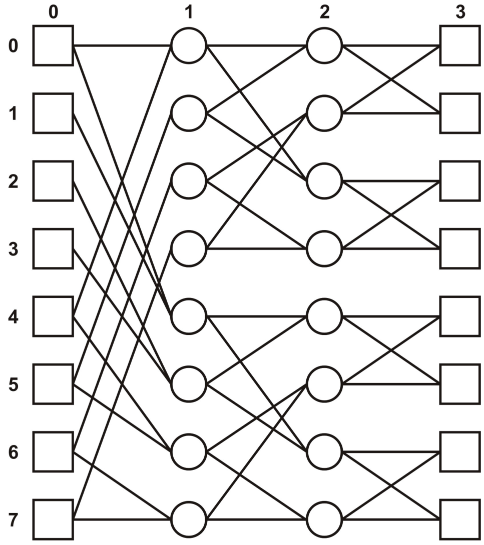


Figure 13. Banyan.

4. Self-Similarity in Parallel Algorithms

There are many parallel algorithms that have self-similarity characteristics. We will present some of them: compression on a tree and a hypercube and all-to-all communication on a hypercube. The Cooley–Tukey algorithm for fast Fourier transform (FFT) and the bitonic merge sort algorithm are other significant examples in this respect. We will renounce giving details about them, because their implementation on a hypercube shows very easy self-similarity.

4.1. Compression on a Tree

Definition 10. *Let us consider a set M of $n = 2^m$ elements, $M = \{a_i / i = 0, 1, \dots, n-1\} \subseteq M_r$ = the reference set. The set M follows being processed for calculating the value $a_1 \oplus \cdots \oplus a_n$, where $\oplus$ is an associative algebraic operation defined on M_r.*

The reference set M_r could be $\mathbb{R}$, and $\oplus$ may be $+$, min, max, etc. The pseudocode Algorithm 1 describes the compression procedure on a tree.

Algorithm 1 Compression on a tree.

```
Notations:
- $A[0...2n-1]$ is a unidimensional array of size $2n = 2^{m+1}$.
Premise:
- The input data are stored in the array $A[0...2n-1]$, at the locations $A[n], A[n+1] \dots, A[2n-1]$.
Algorithm:
for $k \leftarrow m-1$ down to 0 do
    for all $j$ : $2^k \leq j \leq 2^{k+1}-1$ in parallel do
        $A[j] \leftarrow A[2j] \oplus A[2j+1]$
    end for
end for
```

An example is presented in Figure 14.

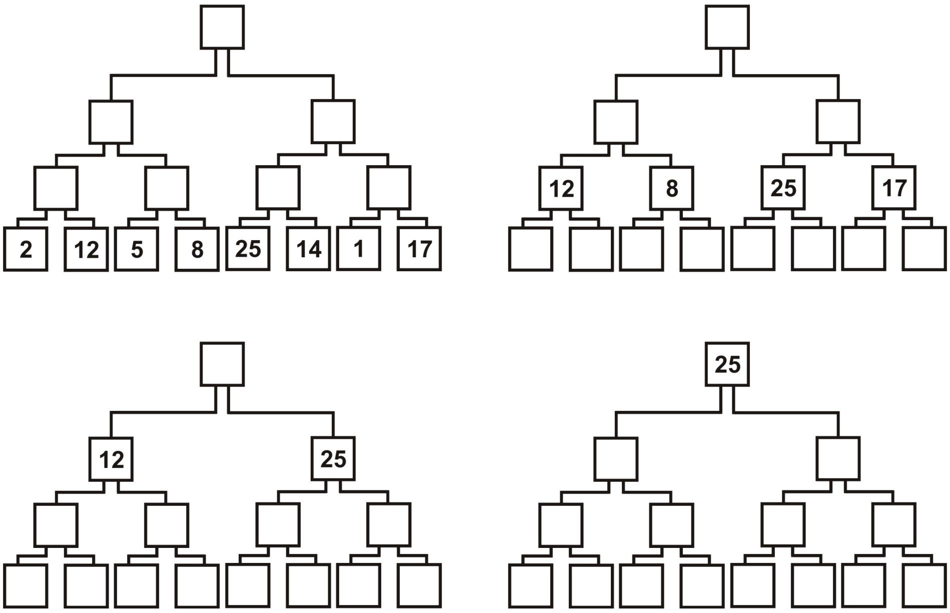


Figure 14. Maximum of eight integers on a full and complete binary tree.

The evolution of the computation is similar to the reverse of the Cantor set building procedure. In fact, the construction of the Cantor set corresponds to the first phase of the divide-and-conquer paradigm, the same as the compression recursive algorithm. This explains such fitness.

4.2. Compression on Hypercube

The pseudocode Algorithm 2 describes the compression procedure on the hypercube [7]. An example is presented in Figure 15.

Algorithm 2 Summing $n = 2^m$ integers on a hypercube with $p = 2^q$ vertices (q dimensions, $q < m$): pseudocode for the processing unit located at vertex i.

```
Notations:
- $A$ is a unidimensional array of size $n = 2^m$.
- $H$ is a hypercube with $p = 2^q$ vertices.
- $q$ is the number of $H$ hypercube dimensions, $q < m$.
- $s$ is an integer variable that stores the final sum, computed by processing unit located at vertex 0.
- $s_r$ is an integer variable that stores the partial sum received by processing unit located at vertex $i$ from a neighbor.
Premise:
- Initially, the number sequence is stored in the array $A$.
Algorithm:
for $j \leftarrow 0$ to $2^{m-q} - 1$ do
  $s \leftarrow s + A[i * 2^{m-q} + j]$
end for
$mask \leftarrow 0$
for $j \leftarrow 0$ to $q - 1$ do
  if $(i$ and $mask) = 0$ then
    if $(i$ and $2^j) = 0$ then
      $source \leftarrow i$ xor $2^j$
      receive $s_r$ from $source$
      $s \leftarrow s + s_r$
    else
      $destination \leftarrow i$ xor $2^j$
      send $s$ to $destination$
    end if
  end if
  $mask \leftarrow mask$ xor $2^j$
end for
```

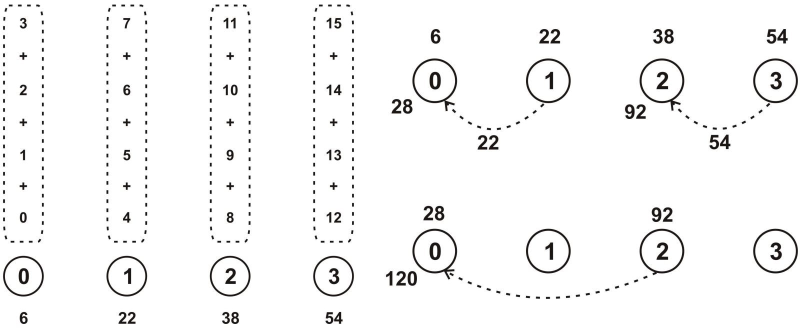


Figure 15. Summing 16 integers on a hypercube with four vertices.

Again, the evolution of the computation is similar to the reverse of the Cantor set building procedure.

4.3. All-To-All Diffusion on a Hypercube

Definition 11. *Each processing unit of p_i sends a message to all others.*

Algorithm 3 describes all-to-all diffusion on a hypercube [7]. An example is presented in Figure 16.

Algorithm 3 All-to-all communication on a hypercube: pseudocode for a processing unit located at vertex i.

```
Notations:
- $H$ is a hypercube with $p = 2^q$ vertices.
- $q$ is the number of $H$ hypercube dimensions.
Premise:
Initially, the processing unit $p_i$ from the vertex $i$ has in its own memory a message $m_i$
Algorithm:
$collected\_messages \leftarrow m_i$
for $j \leftarrow 0$ to $q - 1$ do
    $partner \leftarrow i$ xor $2^j$
    send $collected\_messages$ to $partner$
    receive $message$ from $partner$
    $collected\_messages \leftarrow collected\_messages \cup message$
end for
```

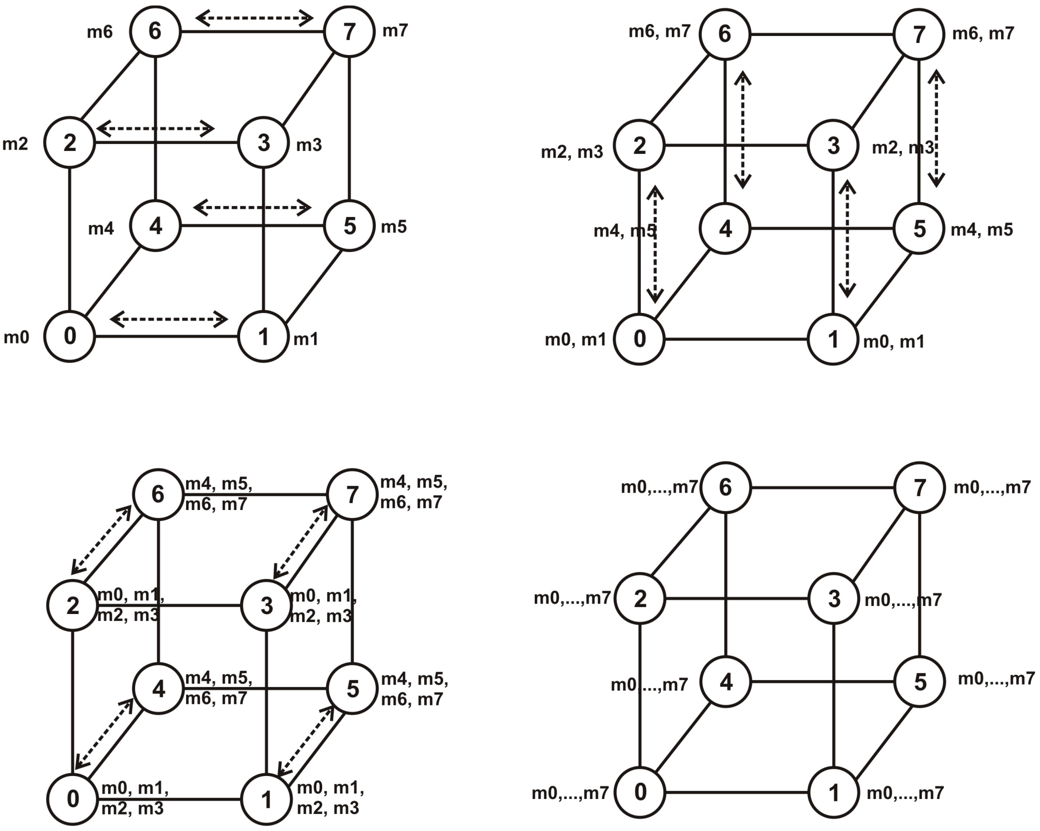


Figure 16. Example of all-to-all communication on a three-dimensional hypercube.

The computation pattern has a self-similarity property. At the k-th iteration, the processing units from a k-dimensional hypercube exchange information with the processing units from the corresponding k-dimensional hypercube.

5. Incursion in Parallel Communication Topologies and in Parallel Algorithms

5.1. Anticipatory Systems

The concept of an anticipatory system was proposed for the first time by Robert Rosen in [8]: "a system containing a predictive model of itself and/or of its environment, which allows it to state at an instant in accord with the model's predictions pertaining to a later instant". The final causation of Aristotle stands at the base of such systems. The future influences the present time. In this way, the causality principle seems reversed.

For example, organizing an event takes into account the past similar events, the present context and a range of information posterior preparation phase: number of participants, weather, events in the chosen location and many other unknown factors before the actual conduct of the event.

> In general, any human action at each current time takes into account the past events, the current situation in the environment, and the future anticipated events. The anticipation in human actions deals with conscious and intentionality, a self-referential finality.
>
> —Daniel M. Dubois [9]

As is well known, Aristotle's causation system has four components: material cause, formal cause, efficient cause and final cause. At present, it is considered that modem physics and mechanics only deal with efficient cause and biology with material cause [10]. Robert Rosen considers that the first three components of Aristotle's causal system are included in the Newtonian formalism, but "the introduction of a notion of final cause into the Newtonian picture would amount to allowing a future state or future environment to cause change of state in the present, and this would be incompatible with the whole Newtonian picture. This is one of the main reasons that the concept of Aristotelian finality is considered incompatible with modern science. In modern physics, Aristotelian ideas of causality are confused with determinism, which is quite different. That is, determinism is merely a mathematical statement of functional dependence or linkage. As Russell points out, such mathematical relations, in themselves, carry no hint as to which of their variables are dependent and which are independent" [11].

The final cause could affect the present state of evolving systems. For this reason, the classical mathematical models are unable to explain many of these biological systems. Furthermore, the final cause seems to be essential for physical and computational systems [10].

5.2. Recursion, Incursion and Hyperincursion

An incursion is an extension of recursion [9,12,13]:

$$x(t+1) = f[\ldots, x(t-2), x(t-1), x(t), x(t+1), \ldots, p] \tag{9}$$

The new state of the variable x depends on the past and/or present states, but also on future states. A particular case is given by Equation (10):

$$x(t+1) = f[x(t), x(t+1), p] \tag{10}$$

The value of the variable x depends on the value of this variable at the preceding time step t, but also at time $t+1$. This dependence is given by the function f. Equation (10) can be associated with a self-referential system, which is an anticipatory system. By replacing the right occurrence of $x(t+1)$ by $f[x(t), x(t+1), p]$, we obtain:

$$x(t+1) = f[x(t), f[x(t), x(t+1), p], p] \tag{11}$$

Now, the associated system explicitly contains a predictive model of itself. This inclusion can continue indefinitely. This paradox can be solved in the following manner: such an anticipatory system has fixed points, which represent its implicit finality. Contrary to the control theory, the goal of this anticipatory system is not imposed from outside, but by the system itself.

The above anticipatory system is based on a composition principle. Containing the model of itself, it evolves by assembling past states based on the model.

The hyperincursion is an incursion where there are multiple potential future states at each time step [9].

Let us consider the following equation:

$$x(t) = ax(t+1)[1 - x(t+1)] \tag{12}$$

Equation (12) conducts to the hyper-recursive equation:

$$x(t+1) = 1/2[1 \pm \sqrt{[1 - 4x(t)/a]}] \tag{13}$$

There are two solutions at each time step. If the system selects itself as one of the two alternatives, a self-organizing anticipatory system is defined. Without selection, this system will store in itself all of the potential solutions. The immune systems work in a similar manner.

5.3. *Incursion in Parallel Communication Topologies*

Dubois proposed in [10] a hyperincursive fractal machine, that is a cellular automaton with incursive sequential computations based on exclusive OR logic. A cell state is computed at the future time $t + 1$ as a function of its neighbors at the present and/or past time steps, but also at the future time step $t + 1$. The incursion becomes a hyperincursion when there are multiple possible future states at each time step. Dubois's hyperincursive fractal machine shows fractal pattern generation and quantum effects [10].

The analysis of Dubois's fractal machine conducts us to an important observation: the incursion is natural in pipeline computation. Thus, parallel computing systems that use communication topologies that permits pipeline computation are a natural habitat for solving incursive equations. Full and complete binary tree, fat tree, mesh, hypercube, butterfly and banyan are such communication topologies.

5.4. *Incursion in the Floyd–Warshall Algorithm*

Many parallel algorithms, especially those based on dynamic programming technique, are suitable for incursive computation. We have chosen for demonstration the Floyd–Warshall algorithm, which is the base of the famous Algebraic Path Problem (APP).

5.4.1. Floyd–Warshall Algorithm

We consider a weighted digraph $(G, \ell) = (\langle V, A\rangle, \ell)$. The problem resides in determining, for any two nodes i, j, a path of minimum length from i to j (if there is one). The used method is dynamic programming [14].

We extend the function ℓ to $\ell : V \times V \to \mathrm{R}$, by assigning $\ell_{ij} = \infty$ for those pairs of distinct nodes where $\langle i, j\rangle \notin A$ and $\ell_{ii} = 0$ for every $i = 0, \ldots, n-1$.

Let us define the state of the problem as being the sub-problem corresponding to the determination of the minimum length paths with intermediary nodes from the set $X \subseteq V$, MP2ND(X) (Minimum Path between any two Nodes of a Digraph). Obviously, MP2ND(V) is the initial problem.

The minimum path from i to j built with intermediary nodes from X will be denoted by ℓ_{ij}^{X}. If $X = \emptyset$, then $\ell_{ij}^{\emptyset} = \ell_{ij}$.

Let us consider the optimal decision that transforms the state MP2ND$(X \cup \{k\})$ to MP2ND(X). We assume that (G, ℓ) is a weighted digraph without negative cycles. Let ρ be an optimum path from i to j that contains intermediary nodes from the set $X \cup \{k\}$. We have length$(\rho) = \ell_{ij}^{X \cup \{k\}}$, where length$(\rho)$ is the length of the path ρ. If the node k does not belong to ρ, then the policy of obtaining ρ corresponds to the state MP2ND(X), and by applying the principle of optimality, we obtain:

$$\ell_{ij}^{X} = \text{length}(\rho) = \ell_{ij}^{X \cup \{k\}} \tag{14}$$

In the case when k belongs to the path ρ, we consider ρ_1 the sub-path of ρ from i to k and ρ_2 the sub-path from k to j. The intermediate nodes on the two sub-paths are only from X.

According to the principle of optimality, the optimal policy corresponding to the state MP2ND(X) is a sub-policy of the optimal policy corresponding to the state MP2ND$(X \cup \{k\})$. It results that ρ_1 and ρ_2 are optimal in MP2ND(X). From here, it results:

$$\ell_{ij}^{X \cup \{k\}} = \text{length}(\rho) = \text{length}(\rho_1) + \text{length}(\rho_2) = \ell_{ik}^{X} + \ell_{kj}^{X} \tag{15}$$

The functional analytical equation for the optimal values ℓ_{ij}^{X} is:

$$\ell_{ij}^{X \cup \{k\}} = \min\{\ell_{ij}^{X}, \ell_{ik}^{X} + \ell_{kj}^{X}\} \tag{16}$$

Lemma 1. *If (G, ℓ) does not have negative length cycles, we have the following relations:*

$$\ell_{kk}^{X \cup \{k\}} = 0 \tag{17a}$$

$$\ell_{ik}^{X \cup \{k\}} = \ell_{ik}^{X} \tag{17b}$$

$$\ell_{kj}^{X \cup \{k\}} = \ell_{kj}^{X} \tag{17c}$$

for every $i, j, k \in V$.

By means of Lemma 1, the relation (16) becomes:

$$\ell_{ij}^{X \cup \{k\}} = \min\{\ell_{ij}^{X}, \ell_{ik}^{X \cup \{k\}} + \ell_{kj}^{X \cup \{k\}}\} \tag{18}$$

This is an incursive equation.

Without loosing of generality, we can consider $X = \{0, 1, \ldots, k-1\}$ and $X \cup \{k\} = \{0, 1, \ldots, k-1, k\}$. The computing of optimal values results from the solving of sub-problems:

$$\text{MP2ND}(\emptyset), \text{MP2ND}(\{0\}), \text{MP2ND}(\{0,1\}),$$
$$\ldots, \text{MP2ND}(\{0, 1, \ldots, n-1\}) = \text{MP2ND}(V)$$

Using the notation ℓ_{ij}^{k} for $\ell_{ij}^{\{0,\ldots,k\}}$, Relation (18) becomes:

$$\ell_{ij}^{k} = \min\{\ell_{ij}^{k-1}, \ell_{ik}^{k} + \ell_{kj}^{k}\} \tag{19}$$

The incursive relation (19) is the heart of the Floyd–Warshall algorithm.

5.4.2. Algebraic Path Problem

The algebraic path problem unifies the strategies used in solving three major class problems, being independently developed, with its own algorithms:

- Determining the minimum paths in a graph with the Floyd–Warshall algorithm;
- Determining the transitive closure of a partial order relation;
- Solving the linear equation systems with the Gauss–Jordan method.

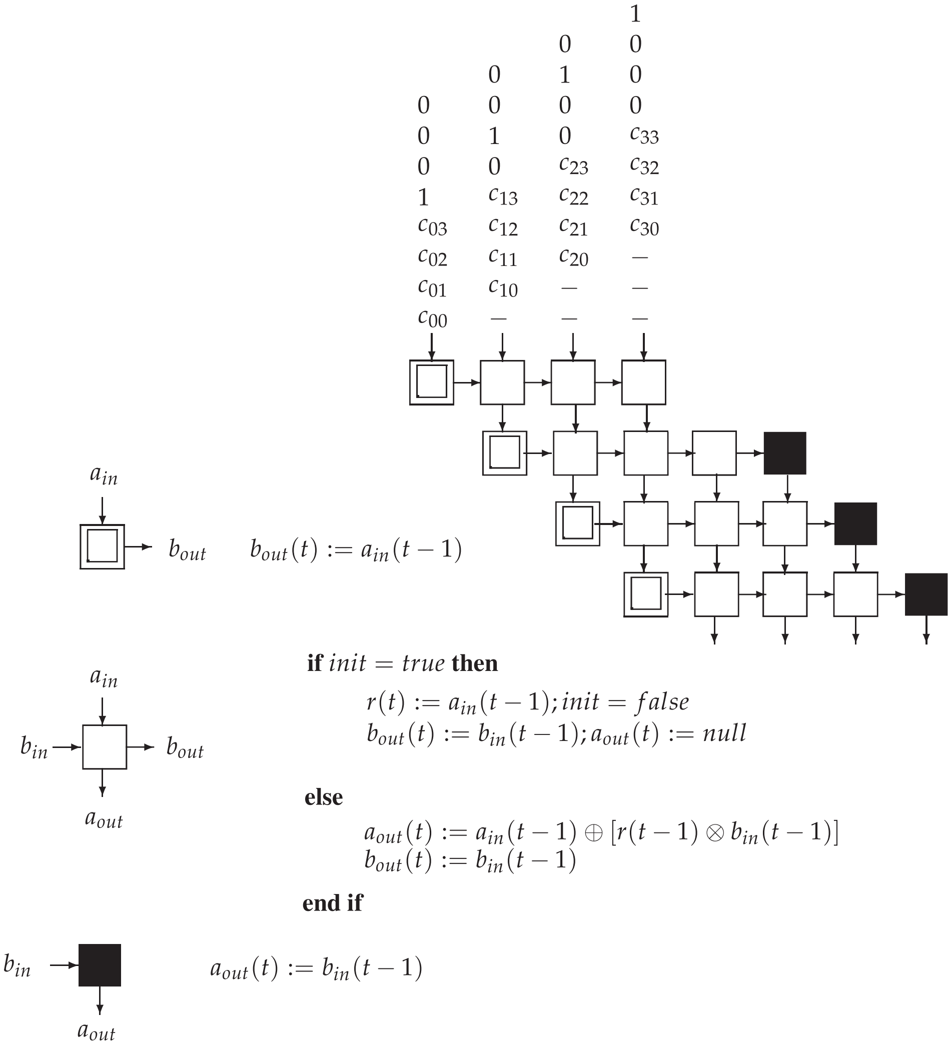
Definition 12. *Given a weighted graph $G = (V, E, w)$, where $V = \{1, 2, \ldots, n\}$, $w : E \to H$ and $(H, \oplus, \otimes)$ forms a unit ring, determine the matrix $C_{n \times n}$, so that:*

$$C[i, j] = \bigoplus_{\substack{p\ path \\ from\ i\ to\ j}} w(p)$$

If $p = i_0 i_1 ... i_{k-1} i_k$, then $w(p) = \bigotimes_{l=0}^{k-1} w[i_l, i_{l+1}]$.

The incursive relation from the Floyd–Warshall algorithm can be generalized according to the algebraic path problem definition.

There are multiple systolic approaches of the algebraic problem of paths. Noteworthy is the implementation of Kleen's algorithm on a connected-mesh network given by Y. Robert and D. Trystram [15]. See Figure 17.



$b_{out}(t) := a_{in}(t-1)$

```
if $init = true$ then
    $r(t) := a_{in}(t-1); init = false$
    $b_{out}(t) := b_{in}(t-1); a_{out}(t) := null$
else
    $a_{out}(t) := a_{in}(t-1) \oplus [r(t-1) \otimes b_{in}(t-1)]$
    $b_{out}(t) := b_{in}(t-1)$
end if
```

$a_{out}(t) := b_{in}(t-1)$

Figure 17. Systolic network of Y. Robert and D. Trystram.

6. Conclusions and Future Work

Classical computers work under the classical physical laws, while quantum computers apply quantum physics rules. They are placed at the macroscopic and the atomic scale, respectively. A fractal medium could simultaneously host computations of both types (classical and quantum), because of the scale factor included in its definition, which induces self-similarity, meaning that the part is the same as the whole.

In this context, the most important contributions of this paper are the following:

1. We have proven that the most important parallel communication topologies and many parallel algorithms have fractal properties. As a consequence, plausible computing on a fractal medium could include classical parallel computing.
2. We have identified several parallel algorithms that are suitable for incursive and hyperincursive computation. This opens strong relations among parallel computing, quantum computing and fractal computing.

We intend to continue our investigation on a presumable fractal computer by defining a detailed model, extending the fractal logic and simulating some algorithms.

Author Contributions: Mitică Craus and Vlad-Sergiu Bîrlescu have contributed to all sections of this paper. Maricel Agop contributed to preparing Section 1, and he conceived of Section 2 (Self-Similarity Property of Fractals). All the authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bîrlescu, V.S.; Agop, M.; Craus, M. Computational Properties of a Fractal Medium. *Int. J. Quantum Inf.* **2014**, *12*, 1450022.
2. Holland, P.R. *The Quantum Theory of Motion*; Cambridge University Press: Cambridge, UK, 1993.
3. Agop, M.; Botez, I.; Casian, I.; Birlescu, V.S.; Popa, R.F. On the Memorizing Ability of Nanostructures. *J. Comput. Theor. Nanosci.* **2015**, *12*, 682–688.
4. Craus, M.; Bîrlescu, V.; Agop, M. High Performance Computing—A New Perspective through Fractal Computers (I). *Bull. Polytech. Inst. Iasi Math. Theor. Mech. Phys. Sect.* **2011**, *57*, 24–36.
5. Mandelbrot, B.B. *The Fractal Geometry of Nature*; W.H. Freeman: New York, NY, USA, 1983.
6. Gottlieb, I.; Mociuţchi, C.; Bîrlescu, V.; Craus, M.; Magop, D.; Agop, M. High Performance Computing—A New Perspective through Fractal Computers (III): Elements of Fractal Theory. *Bull. Polytech. Inst. Iasi Math. Theor. Mech. Phys. Sect.* **2011**, *57*, 47–66.
7. Kumar, V.; Grama, A.; Gupta, A.; Karypis, G. *Introduction to Parallel Computing: Design and Analysis of Algorithms*; Addison Wesley: Boston, MA, USA, 2013.
8. Rosen, R. *Anticipatory Systems*; Pergamon Press: Oxford, UK, 1985.
9. Dubois, D.M. Introduction to Computing Anticipatory Systems. *Int. J. Comput. Anticip. Syst.* **1998**, *2*, 3–14.
10. Dubois, D.M. Hyperincursive Methods for Generating Fractals in Automata Related to Diffusion and Wave Equations. *Int. J. Gen. Syst.* **1998**, *27*, 141–180.
11. Rosen, R. Causal Structure in Brain and Machines. *Inf. J. Gen. Syst.* **1986**, *12*, 107–126.
12. Dubois, D.M. Introduction of the Aristotle's final causation in CAST: Concept and method of incursion and hyperincursion. In *Computer Aided Systems Theory—EUROCAST'95*, Proceedings of the Fifth International Workshop on Computer Aided Systems Theory, Innsbruck, Austria, 22–25 May 1995; Pichler, F., Moreno Diaz, R., Albrecht, R., Eds.; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1030; pp. 477–493.
13. Dubois, D.M. A semantic logic for CAST related to Zuse, Deutsch and McCulloch and Pitts computing principles. In *Computer Aided Systems Theory—EUROCAST'95*; Proceedings of the Fifth International Workshop on Computer Aided Systems Theory, Innsbruck, Austria, 22–25 May 1995; Pichler, F., Moreno Diaz, R., Albrecht, R., Eds.; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1030; pp. 494–510.
14. Lucanu, D.; Craus, M. *Proiectarea Algoritmilor*; Polirom Press: Iasi, Romania, 2008.

15. Robert, Y.; Trystram, D. Parallel Implementation of the Algebraic Path Problem. In *CONPAR 86*, Proceedings of the Conference on Algorithms and Hardware for Parallel Processing, Aachen, Germany, 17–19 September 1986; Händler, W., Haupt, D., Jeltsch, R., Juling, W., Lange, O., Eds.; Springer: Berlin/Heidelberg, Germany, 1986; Volume 237; pp. 149–156.