Theory of Solid Matrices & Solid Latrices,

Introducing New Data Structures MA, MT:

for Big Data

Ranjit Biswas

Department of Computer Science, Jamia Hamdard University
Hamdard Nagar, New Delhi–110062, India
ranjitbiswas@yahoo.com

Copyright © 2013 Ranjit Biswas. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Matrix theory has an unlimited volume of applications in all branches of Science, Engineering, Statistics, Optimization, Numerical Analysis, Computer Science, Medical Science, Economics, etc. The present day world dealing with big data (expanding very fast in 4Vs: Volume, Variety, Velocity and Veracity) needs advanced kind of logical and physical storage structures, advanced kind of heterogeneous data structures, new mathematical theories and new models, all these four together we call by 4Ns. In this paper the author introduces the notion of ‘solid matrix’ i.e. n-dimensional hyper-matrix, as a generalization of the classical matrix, and also the notion of ‘solid latrix’. Consequently the author develops the ‘Theory of Solid Matrices’ explaining various operation, properties and propositions on them. Then the author introduces two powerful data structures: a heterogeneous data structure MA and a homogeneous data structure MT. As a very important application of MA and MT, the author proposes a method on how to implement Solid Matrices, n-dimensional arrays, n-dimensional larrays etc. in a computer memory. It is claimed that the new powerful data structures MT and MA can play very important roles as powerful tools in the processing of big data (if arranged as a SM/SL), big temporal databases, etc. to the present data-dependent giant galaxies of organizations, institutions and individuals. It is also claimed that the ‘Theory of Matrices/Latrices’ will be useful in all branches of Sciences, in all branches of Engineering & Technology, Statistics, OR, Cosmology, Medical Sciences, etc. to list a few only out of many, for making better study and analysis if based upon complex computing.
**Mathematics Subject Classification:** 11C20, 15B36

**Keywords:** Big data, 4V-set, 4N-set, solid matrix, solid latrix, semi-cube, cube, larray, r-Train, r-Atrain, MT, MA.

### 1 INTRODUCTION

The present world of big data [1-6,10] are expanding very fast in 4Vs: Volume, Varity, Velocity and Veracity, and also in many more directions. How to deal with big data, how to process big data in an efficient way within limited resources, etc. are of major concern to the computer scientists now-a-days. In particular, the ‘Velocity’ at which the big data have been expanding (or, the 4Vs in which big data have been expanding very fast in the present day world) does not have a one-to-one matching with the ‘Velocity’ at which the new hardware or new software or new mathematical theories or new models are being developed by the scientists. Let us designate the following two sets by 4V-set and 4N-set:

(i) 4V-set = { Volume, Varity, Velocity and Veracity}, and
(ii) 4N-set = {New Theories, New Hardware, New Software, New Models}.

It is obvious that big data can be efficiently processed by a faster development of the 4N-set only. If 4V-set continues its dominance over 4N-set with respect to time, then it will be difficult to the world to think of “BIG DATA : A Revolution That Will Transform How We Live, Work, and Think” [14]. As on today, the 3N-set lagging behind in the race with 4V-set.

In this paper the author works on two elements of the 3N-set. First of all the author introduces the notion of ‘Solid Matrix’ and of ‘Solid Latrix’, and then develops the ‘Theory of Solid Matrices’. A solid matrix can be regarded as a mathematical object which can facilitate operations on big data in many cases. Multidimensional structure [4] is quite popular for analytical databases that use online analytical processing (OLAP) applications. Analytical databases use these databases because of their ability to deliver answers to complex business queries.
swiftly. Data can be viewed from different angles, which gives a broader perspective of a problem unlike other models. Then the author introduces two new powerful data structures MT and MA which can deal with big data. Finally some applications of the data structures MT and MA in some categories of big data are discussed. The notion of multi-dimensional matrices studied by Ashu M. G. Solo in [1] (and also by Christian Krattenthaler et. al. in [2]) is almost analogous to our notion of solid matrices; but our ‘Theory of Solid Matrices’ is precise, complete and sound, compatible and scalable with the real life application domains of any organization, easy to implement in computer memory and easy to be applied in the real problems of various fields of Science, Engineering, Statistics, OR, etc.

2 SOLID MATRIX & SOLID LATRAX

We know that a matrix is a rectangular array of numbers or other mathematical objects, for which various operations such as addition and multiplication are defined [6]. Most commonly, a matrix over a field F is a rectangular array of scalars from F. But we may also consider a generalized kind of matrix whose elements are objects [13] over the region RR, or over any appropriate region R. For details about the Region Algebra and the ‘Theory of Objects’ in a region, one could see Biswas [13].

We define a solid matrix as an n-dimensional hyper-matrix where n > 2 and the elements are objects from the region RR or from any appropriate region R [13], none being $\varepsilon$ elements [11,12]. We say that it has n number of hyper layers. However, a solid matrix is a mathematical object and should not be confused with the data structure ‘n-dimensional array’ in computer science. For details about the n-dimensional array (multi-dimensional array) and its MATLAB implementation, one could see any good MATLAB book.

We define a lattice as a rectangular array of numbers (or, objects from the region RR or from any appropriate region R) and $\varepsilon$ elements [11,12]. We define a solid lattice as an n-dimensional hyper-lattice where n > 2 and the elements are objects from the region RR or from an appropriate region R [13],
and may be $\varepsilon$ elements [11,12].

We use the notation $n$-$SM$ to denote an $n$-dimensional solid matrix and $n$-$SL$ to denote an $n$-dimensional solid latrix. An abstract bottom-up approach to view the structure of an $n$-$SM$ is given below:

Imagine a classical two dimensional matrix $S_2$ of size $m_1 \times m_2$. Suppose that the matrix $S_2$ expands in a new dimension upto a height $m_2$ to form a 3-$SM$ $S_3$. Now suppose that the matrix $S_3$ expands in another new dimension upto a height $m_3$ to form a 4-$SM$ $S_4$ and so on. Finally, suppose that the matrix $S_{n-1}$ expands in another new dimension upto a height $m_n$ to form a $n$-$SM$ $S_n$. Thus we have an $n$-$SM$ $S_n$ of size $m_1 \times m_2 \times m_3 \times \ldots \times m_n$ where $n \geq 3$.

2.1 Height of a Solid Matrix (Solid Latrix)

Consider an $n$-$SM$ ($n$-$SL$) $S$ of size $m_1 \times m_2 \times m_3 \times \ldots \times m_n$ where $n \geq 3$. The last suffix-index $m_n$ is called the **height** of the solid matrix (solid latrix) $S$. We write it as $\text{height}(S) = m_n$. As a trivial case, we assume that the height of a classical matrix/latrix (i.e. of a two dimensional matrix/latrix) is 1.

2.2 Base Size, Base Matrix and Base Latrix

Consider an $n$-$SM$ ($n$-$SL$) $S$ of size $m_1 \times m_2 \times m_3 \times \ldots \times m_n$ where $n \geq 3$. Then the **base size** of $S$ is $m_1 \times m_2$ which is constituted by the first two indices $m_1$ and $m_2$ out of the size of the SM (SL) $S$. The sub-matrix (sub-latrix) $B$ of the SM (SL) $S$ of size $m_1 \times m_2 \times m_3 \times m_4 \times \ldots \times m_n$ where $m_3 = m_4 = \ldots = m_n = 1$ is called the **base matrix** (base latrix) of $S$. As a trivial case, we assume that the base size of a classical matrix/latrix $M$ (i.e. of a two dimensional matrix/latrix) is the size of the matrix/latrix itself, and the base matrix (base latrix) of $M$ is $M$ itself.

There are many real life examples of 3-D solid matrices/latrices in different branches of science and engineering. We, first of all, present a detailed algebraic study of 3-D solid matrices/latrices. We explain the structure of 3-D solid matrices/latrices, study various operations on them and various properties of them.

The same can be well extended for $n$-dimensional solid matrices (hyper matrices) or for $n$-dimensional solid latrices (hyper latrices) in an analogous way.
2.3 3-D Solid Matrix (3-SM) & Some Characterizations

Consider \( h \) number of \( m \times n \) matrices \( M_1, M_2, M_3, \ldots, M_h \) where \( m, n, h \) are positive integers. Consider an abstract multilayer structure where layer-1 is the bottom most layer which is the matrix \( M_1 \), layer-2 is the matrix \( M_2 \), layer-3 is the matrix \( M_3 \), \ldots, layer-\( h \) is the top most layer matrix \( M_h \). One could view this structure as a rectangular parallelepiped from distance. This rectangular parallelepiped is called a 3-D Solid Matrix (in short, 3-SM) of size \( m \times n \times h \). A solid matrix \( S \) is thus a 3-dimensional matrix having \( mnh \) number of cells. We say that the length (row-size) of the SM \( S \) is \( m \), the breadth (column-size) of the SM \( S \) is \( n \) and the height of the SM \( S \) is \( h \). We denote a 3-SM \( S \) of size \( m \times n \times h \) as below:

\[
S = < M_1, M_2, M_3, \ldots, M_h >.
\]

If \( m = n = h \) (\( = a \), say), then the SM is called a \textbf{Cube} of side \( a \), which is having \( a^3 \) number of cells. If \( m = n \), then the SM is called a \textbf{Semi-cube}. Each layer of a cube or semi-cube is a square matrix. For an \( n \)-SM \( S \) of size \( m_1 \times m_2 \times m_3 \times \ldots \times m_n \) where \( n \geq 3 \), if \( m_1 = m_2 = m_3 = \ldots = m_n \) (\( = a \), say) then the SM is called a \textbf{n-dimensional Hyper Cube} of size \( a \).

![Figure-1: A semi cube of height \( h \).](image)

**Example of a solid matrix**

Consider a class of 3\textsuperscript{rd} semester MS(CS) Programme in a university having 60
students in the class who are $X_1, X_2, X_3, \ldots, X_{60}$. There are five number of ‘Theory’ courses in this semester which are $C_1, C_2, C_3, C_4,$ and $C_5$ being taught by five lecturers. Every weak one Sessional Examination is to be conducted over 50 marks by every lecturer on the subject he is teaching, as per university bye-laws for ‘Continuous Evaluation Systems’. In total 12 number of Sessional Examination are to be conducted on each course in this semester. Clearly, the result of students in 1st Sessional Examination form the matrix $M_1$ of size $5 \times 60$ as shown below:

$$
\begin{array}{cccc}
X_1 & X_2 & \cdots & X_{60} \\
C_1 & 42 & 18 & \cdots & 49 \\
C_2 & 37 & 25 & \cdots & 08 \\
C_3 & 49 & 50 & \cdots & 42 \\
C_4 & 02 & 00 & \cdots & 05 \\
C_5 & 39 & 28 & \cdots & 50 \\
\end{array}
$$

Figure-2: Layer-1 (bottom layer) matrix $M_1$ of a 3-SM

In a similar way there will be eleven matrix more, each of size $5 \times 60$. These twelve matrices $M_1, M_2, M_3, \ldots, M_{12}$, if logically placed one above another, will form one solid matrix (3-SM) $S$ of size $5 \times 60 \times 12$. If there is a query: “What is the marks obtained by the student $X_29$ in the $8^{th}$ Sessional Examination in course $C_3$?”, the answer will be the data element $s_{3,29,8}$ of S.

2.4 Null Solid Matrix

The 3-SM $S = < O_{m \times n}, O_{m \times n}, O_{m \times n}, \ldots, O_{m \times n} >$ of size $m \times n \times h$, where $O_{m \times n}$ is the classical null matrix of size $m \times n$, is called the Null 3-SM of size $m \times n \times h$.

In a recursive way the above definition can be extended to define a Null $n$-SM as below:

The $n$-SM $S_0 = < O_{m_1 \times m_2 \times m_3 \times \ldots \times m_{n-1}}, O_{m_1 \times m_2 \times m_3 \times \ldots \times m_{n-1}}, \ldots >$.
O_{m_1 \times m_2 \times m_3 \times \ldots \times m_{n-1}} \text{ of size } m_1 \times m_2 \times m_3 \times \ldots \times m_n \text{ where } O_{m_1 \times m_2 \times m_3 \times \ldots \times m_{n-1}} \text{ is the null (n-1)-SM of size } m_1 \times m_2 \times m_3 \times \ldots \times m_{n-1} \text{ is called the \textbf{Null n-SM of size } m_1 \times m_2 \times m_3 \times \ldots \times m_n.}

\subsection{2.5 Unit Semi-Cube & Unit Cube}

The semi-cube \( I = < I_{m \times m}, I_{m \times m}, I_{m \times m}, \ldots, I_{m \times m} > \text{ of size } m \times m \times h, \text{ where } I_{m \times m} \text{ is the classical unit matrix of order } m \times m, \text{ is called the \textbf{Unit Semi-cube of size } m \times m \times h.} \) The cube \( I = < I_{m \times m}, I_{m \times m}, I_{m \times m}, \ldots, I_{m \times m} > \text{ of size } m \times m \times m, \text{ where } I_{m \times m} \text{ is the classical unit matrix of order } m \times m, \text{ is called the \textbf{Unit Cube of size } m \times m \times m.} \) In a recursive way the above definition can be extended to define a ‘\textbf{Unit Hyper Semi-Cube}’ and ‘\textbf{a Unit Hyper Cube}’.

\section{3 ALGEBRA OF SOLID MATRICES}

In this section we present few basic operations on SMs and their properties.

\subsection{3.1 Addition/Subtraction of two SMs}

Two SMs can be added (or subtracted) if they are of same size, and the resultant SM is also of the same size. Consider two SMs \( S_1 \) and \( S_2 \), each of size \( m \times n \times h \), given by

\[ S_1 = < M_{11}, M_{12}, M_{13}, \ldots, M_{1h} > \quad \text{and} \quad S_2 = < M_{21}, M_{22}, M_{23}, \ldots, M_{2h} >. \]

Then

\[ S_1 + S_2 = < M_{11} + M_{21}, M_{12} + M_{22}, \ldots, M_{1h} + M_{2h} > \quad \text{and} \]

\[ S_1 - S_2 = < M_{11} - M_{21}, M_{12} - M_{22}, \ldots, M_{1h} - M_{2h} >. \]

\subsection{3.2 Transpose of a SM}

The transpose of a 3-SM \( A = [a_{pqr}] \text{ of size } m \times n \times h \text{ is a 3-SM } B = [b_{uvw}] \text{ of size } n \times m \times h \text{ where } a_{ijk} = b_{jik} \quad \forall \ i, j, k. \text{ We write } B = A^T. \)

Obviously, if \( I \) be a unit semi-cube or a unit cube then \( I^T = I \).

For an n-SM \( A \text{ of size } m_1 \times m_2 \times m_3 \times \ldots \times m_n \), the transpose of \( A \) is defined in a similar way but with respect to a given pair of indices \( i \text{th and } j \text{th}, \text{ where } i \neq j \text{ and } i, j \leq n. \)
Consider an n-SM $A$ of size $m_1 \times m_2 \times m_3 \times \ldots \times m_i \times \ldots \times m_j \times \ldots \times m_n$ and an n-SM $B$ of size $m_1 \times m_2 \times m_3 \times \ldots \times m_j \times \ldots \times m_i \times \ldots \times m_n$. Then the transpose of the n-SM $A$ is the n-SM $B$ with respect to the pair of indices $i$th and $j$th, where $i \neq j$ and $i, j \leq n$, if $a_{pqr\ldots ij\ldots uv} = b_{pqr\ldots ji\ldots uv} \forall p, q, r, \ldots, i, j, \ldots, u, v$. We write $B = A^T$.

**Proposition 3.1**

If $A$ be a n-SM, then $(A^T)^T = A$.

### 3.3 Scalar-height of height $z$

A solid matrix of size $1 \times 1 \times z$ of scalar elements is called a **scalar-height** of height $z$, where $z$ is a positive integer. For example, the 3-SM $H$ below is a scalar-height of height $z$:

$$
H = \begin{pmatrix}
  k_z \\
  k_{z-1} \\
  \vdots \\
  k_2 \\
  k_1
\end{pmatrix}
$$

where $k_1, k_2, k_3, \ldots, k_z$ are scalar quantities.

The scalar-height of height $z$ having all the elements 0 in it is called a **Null Scalar-height** or **Zero Scalar-height** of height $z$ denoted by $O_z$ as shown below:

$$
H = \begin{pmatrix}
  0 \\
  0 \\
  \vdots \\
  0 \\
  0
\end{pmatrix}
$$
Theory of solid matrices & solid latrices

3.4 Scalar-height multiplication of a solid matrix

Consider a 3-SM $S$ of size $m \times n \times h$ and a scalar-height $H$ of height $h$ given by respectively

\[ S = \langle M_1, M_2, M_3, \ldots, M_h \rangle \quad \text{and} \quad H = \begin{bmatrix}
  k_1 & & \\
  k_2 & & \\
  \vdots & \ddots & \\
  k_{h-1} & & k_h
\end{bmatrix} \]

The scalar-height multiplication of $S$ by $H$ yields a SM $M$ given by

\[ M = \langle k_1M_1, k_2M_2, k_3M_3, \ldots, k_hM_h \rangle \]

and we write $M = HS$.

3.5 Scalar multiplication of a solid matrix

Consider a scalar quantity $b$. The scalar multiplication of the 3-SM $S$ of size $m \times n \times h$ by the scalar $b$ is the scalar-height multiplication of $S$ by the scalar-height $H$ (of height $h$, the height of the 3-SM $S$) where
which yields a 3-SM \( \mathbf{M} \) given by \( \mathbf{M} = < bM_1, bM_2, bM_3, \ldots \ldots, bM_h > \)
and we write \( \mathbf{M} = bS \).

### 3.6 Determinant-height of a semi-cube

Consider a semi-cube \( \mathbf{S} \) of size \( m \times m \times h \) given by

\[
\mathbf{S} = < M_1, M_2, M_3, \ldots \ldots, M_h >.
\]

The **determinant-height** (det.ht) of the semi-cube \( \mathbf{S} \) is the scalar-height \( D_S \) given by

\[
D_S = \begin{bmatrix}
|M_1| \\
|M_2| \\
\vdots \\
|M_h| \\
\end{bmatrix}
\]

where \( |M_r| \) is the value of the determinant of the matrix \( M_r \).

If none of the elements of \( D_S \) is 0, then the semi-cube \( \mathbf{S} \) is called to be **Non-Singular** semi-cube. If at least one element in \( D_S \) is 0 then it is called to be **Singular**.

For a unit semi-cube or a unit-cube \( \mathbf{S} \), all the elements in \( D_S \) are 1. Also, it is obvious that for a semi-cube or a cube \( \mathbf{S} \), \( \text{det.ht} (S) = \text{det.ht} (S^T) \).
3.7 Multiplication of two solid matrices

Two 3-SMs $A$ of size $m \times n \times h$ and $B$ of size $r \times s \times t$ are compatible for multiplication $AB$ if:

(i) $n = r$
(ii) $h = t$

If $A = < A_1, A_2, A_3, \ldots, A_h >$ and $B = < B_1, B_2, B_3, \ldots, B_h >$ are two 3-SMs of size $m \times n \times h$ and $n \times s \times h$ respectively, then $AB$ is a 3-SM of size $m \times s \times h$ defined by

$$AB = < A_1B_1, A_2B_2, A_3B_3, \ldots, A_hB_h >.$$ 

Clearly, compatibility of the multiplication $AB$ does not yield the compatibility of the multiplication $BA$. Even if $AB$ and $BA$ both be compatible, they are in general two distinct 3-SMs of two distinct sizes, i.e. the product of two 3-SMs is not commutative. For a semi-cube $S$ of size $m \times m \times h$, the products $S^2, S^3, S^4, \ldots$ etc. are semi-cubes of size $m \times m \times h$. Thus, for a cube $S$ of size $m \times m \times m$, the products $S^2, S^3, S^4, \ldots$ etc. are cubes of size $m \times m \times m$. The SM equation $SM = O$ does not necessarily mean that either $A = O$ or $B = O$ or both.

3.8 Inverse of a Non-Singular Semi-Cube

Inverse of a 3-SM exists if it is a non-singular semi-cube. Consider a non-singular semi-cube $S$ of size $m \times m \times h$ given by

$$S = < M_1, M_2, M_3, \ldots, M_h >.$$ 

Then inverse of $S$ denoted by $S^{-1}$ is a non-singular semi-cube defined by

$$S^{-1} = < M_1^{-1}, M_2^{-1}, M_3^{-1}, \ldots, M_h^{-1} >.$$ 

The following propositions are straightforward and can be easily proved.

**Proposition 3.2**

If $S$ is a non-singular semi-cube of size $m \times m \times h$, then

$$SS^{-1} = S^{-1}S = I \quad \text{(unit semi-cube of size} \ m \times m \times h).$$

**Proposition 3.3**

If $A$ and $B$ are two solid matrices (3-SMs) such that $AB$ exists, then
\[(AB)^T = B^T A^T.\]

**Proposition 3.4**

If \(A\) and \(B\) are two non-singular semi-cubes such that \(AB\) exists, then
\[
(AB)^{-1} = B^{-1} A^{-1}.
\]

**Proposition 3.5**

(i) Associativity holds good i.e. If \(A, B, C\) are three 3-SMs such that \(AB\) and \(BC\) exist, then \(A(BC) = (AB)C\).

(ii) Distributive Property hold good i.e.

\[
\begin{align*}
(a) \quad A (B + C) &= AB + AC \\
(b) \quad (B + C) A &= BA + CA.
\end{align*}
\]

where the multiplications are compatible.

4 HETEROGENEOUS DATA STRUCTURE ‘MA’

Today’s supercomputers or multiprocessor systems which can provide huge parallelism has become the dominant computing platforms (through the proliferation of multi-core processors), and the time has come to stand for highly flexible advanced level of data structures that can be accessed by multiple threads which may actually access large volume of heterogeneous data simultaneously, that can run on different processors simultaneously, even for big data (which are the fast expanding universe in 4Vs : Volume, Varity, Velocity and Veracity) [3,5,7,10,14]. Not much literatures have been so far reported in journals or in publications; however to read about big data initially, the good book [14] may be seen. In most of the giant business organizations, the system has to deal with a large volume of heterogeneous data, heterogeneous type of big data [3,5,7,10,14] for which the data structures of the existing literature can not always lead to the desired solution for thirst or desired optimal satisfaction. The very common and frequent operations like Insertion, deletion, searching, etc. are required to be much and much faster even if the big data [3,5,7,10,14] are of heterogeneous datatypes. Such situations require some way or some method which work more efficiently than the simple rudimentary
existing data structures. Obviously, there is a need of a dash of creativity of a new or better performed heterogeneous data structure for big data which at the same time must be of rudimentary in nature.

In this section we propose a very powerful and dynamic real time heterogeneous data structure MA to deal with big data of heterogeneous datatype, and then we present a very real life generalized application of MA (we call it a ‘generalized application’ in the sense that this application will be useful to all branches of Theoretical and applied Science, in all branches of Engineering & Technology, in all branches of Social Science, Medical Science, Optimization Techniques, in Statistics, in Cosmology, in Astrophysics, etc. etc. to list a few only out of a large number of broad areas). MA is the abbreviation for ‘Multi Atrains’, as it is an extension of the heterogeneous data structure ‘Atrain’ proposed by Biswas in [11]. For details about the properties, operations, algorithms and applications of heterogeneous data structure ‘Atrain’ and of the homogeneous data structure ‘Train’, one could see [11,12].

In the heterogeneous data structure Atrain, there are logically two layers: the pilot is the upper layer and the coaches are in the lower/inner layer. We extend the notion of Atrain by incorporating nil or one or more number of intermediate layers between the pilot (upper layer) and linked-coaches (lower layer) to develop a new heterogeneous data structure ‘Multi Atrains (MA)’. The intermediate layers are usually Atrains, but could be pilots, linked-coaches, or larrays [11,12] too. Type of the various layers, according to the construction-needs for the problems under study, are decided by the developers on behalf of the organization concerned. Thus Atrain may be regarded as a special case of MA, where there is(are) no intermediate layer(s) between the upper layer and the lower layer.

If the total number of layers is called the height, then height(Atrain) = 2, and height(MA) ≥ 2.

In [11,12], two new data structures were proposed by Biswas for storing data while they are huge in volume or even big data. The data structure ‘r-train’ (or ‘train’ in
short) is a homogeneous data structure which can deal with a large volume of homogeneous data very efficiently. The data structure r-atrain (or, atrain) is a robust kind of dynamic heterogeneous data structure. The term ‘Atrain’ stands for “Advanced train’. The datatype in a r-atrain may vary from coach to coach (unlike in r-train), but in a coach all data must be homogeneous i.e. of identical datatype. Thus each coach is homogeneous although the atrain is heterogeneous. Analogous to the construction of MA from the heterogeneous data structure Atrain, we propose the construction of a new data structure MT from the homogeneous data structure Train [11,12]. MT is the abbreviation for ‘Multi Trains’. In the homogeneous data structure Train, there are logically two layers: the pilot is the upper layer and the coaches are in the lower/inner layer. In the data structure MT, there exist nil or one or more number of intermediate layers between the pilot (upper layer) and linked-coaches (lower layer) where the intermediate layers are usually Trains, but could be pilots, linked-coaches, or larrays [11,12] too. Thus Train may be regarded as a special case of the data structure MT, where there is(are) no intermediate layer(s) between the upper layer and the lower layer. Obviously, height(Train) = 2, and height(MT) ≥ 2.

Clearly, an ‘MT of height h’ is a particular case of an ‘MA of height h’.

4.1 Applications of the data structures MA, MT.

The most powerful applications of MT and MA are in the following two broad and important areas (commonly used in all branches of Science, in all branches of Engineering & Technology, etc. etc.):

(i) implementation of solid matrices (solid latrices).
(ii) implementation for multi-dimensional arrays,
(iii) implementation of multi-dimensional larrays [11,12],

where the matrix/latrix elements or array elements or the larray elements are the objects of the region RR or of any appropriate region R [13].

If the data of a particular problem are of heterogeneous datatype, we shall use the
data structure MA, not MT. If all the data are of homogeneous datatype, we shall use the data structure MT (or MA).

4.1.1 Implementation of a 3-SM (3-SL)

About details of implementation of a r-Train in a 8086 memory, one could see [11,12] as a pre-requisite. For implementing a 3-SM (3-SL) of homogeneous data, we use MT of height 3 only. However, in general, for implementation of a n-SM (n-SL) of homogeneous data, we need to use MT of height n. If data are of heterogeneous datatype for many groups where every group is homogeneous in itself, but the datatype of different groups are different in the total database, then we need to use the heterogeneous MA of height n if the SM (SL) is n-dimensional.

We consider here homogeneous data only. Consider a 3-SM (3-SL) $S$ of size $m \times n \times h$ given by $S = < M_1, M_2, M_3, \ldots, M_h >$ of homogeneous data. To implement this 3-SM (3-SL) $S$ in computer memory we need actually one chief Pilot of $h$ number of independent Trains (one for each layer of 3-SM or 3-SL), where each Train is having its own pilot and contains $m$ number of linked coaches. All the $h$ number of trains are independent, but for every train all its coaches are linked/shunted. Surely, we need to consider a MT $M$ with height $= 3$, i.e. three layers in total as shown below:

Upper Layer $L_3$ of the MT $M$:
It is the chief pilot $P = < M_1, M_2, M_3, \ldots, M_h >$ of the MT $M$. The ‘START’ of the MT $M$ points at the chief pilot $P$. This chief pilot $P$ is nothing but a larray [11,12] of $h$ number of elements $M_i$. The element $M_i$ is the address of the $i$th layer of the 3-SM, which is the ‘START’ $S_i$ of the $i$th Train $T_i$ in the MT $M$.

Middle Layer $L_2$ of the MT $M$:
It is the larray [11,12] of $h$ number of pilots corresponding to $h$ number of independent Trains (i.e. $h$ number of independent $r$-Trains where $r = n$ for the present case), given by $< T_1, T_2, T_3, \ldots, T_h >$ where each $T_i$ corresponds to $m$ number of linked/shunted coaches given by :
\[ T_i = \langle C_{i1}, C_{i2}, C_{i3}, \ldots, C_{im} \rangle, \quad i = 1, 2, 3, \ldots, h. \]

At the time of implementation, one has to take care of the ‘status’ of each coach [11,12].

**Lower Layer L₁ of the MT M:**

In this layer, corresponding to each \( i \) there \( m \) number of coaches, and consequently there are in total \( mh \) number of coaches \( C_{ij} \) (\( i = 1, 2, 3, \ldots, h \) and \( j = 1, 2, 3, \ldots, m \)). For a given n-train \( T_i \), each coach \( C_{ij} \) (for \( j = 1, 2, 3, \ldots, m \)) has \( n \) number of passengers [11,12], together with one more (the last one) which is one of the following:

(i) an address to the next coach if “\( j < m \)”, or
(ii) address to the first coach of immediate higher layer if “\( i < h \) and \( j = m \)”, or
(iii) an invalid address \( X \) if “\( i = h \) and \( j = m \).

However, if the data are not all homogeneous, the developer has to go for MA instead of MT.

**Rich Merits of the data structure MT**

From the Chief Pilot \( P \) of the MT, one can directly visit any of the \( h \) number of layer matrices \( M_1, M_2, M_3, \ldots, M_h \) each of which is implemented as an n-Train here. From any n-Train (layer matrix), one can directly visit any of the \( m \) number of rows of this layer matrix each of which is implemented as a coach here. From any coach at the lower layer \( L_1 \), forward journey is possible upto the last coach of the upper layer \( L_h \). Since each coach is a larray, linear visit is possible upto the last element of coach.

**Example**

For the sake of simple presentation here we ignore big size 3-SM, but consider a small size 3-SM \( S = \langle M_1, M_2 \rangle \) of size 3×7×2 of homogeneous data, given by

\[
M_1 = \begin{bmatrix}
2 & 9 & 5 & 8 & 7 & 6 & 6 \\
4 & 6 & 3 & 5 & 4 & 1 & 8 \\
9 & 4 & 0 & 4 & 6 & 2 & 0
\end{bmatrix}
\quad \text{and} \quad
M_2 = \begin{bmatrix}
5 & 8 & 2 & 4 & 0 & 8 & 1 \\
3 & 8 & 1 & 7 & 9 & 3 & 0 \\
8 & 6 & 2 & 7 & 1 & 8 & 5
\end{bmatrix}
\]
For implementing this 3-SM S, we need two 7-Trains \( T_1 \) and \( T_2 \) where each 7-Train will have three coaches, as below:

\[
T_1 = \langle C^1_1, C^1_2, C^1_3 \rangle \quad \text{and} \quad T_2 = \langle C^2_1, C^2_2, C^2_3 \rangle.
\]

It is clear that status \([11,12]\) of each of these six coaches is 0, as there is no \( \varepsilon \) element in this 3-SM. However, it is obvious that for 3-SL, status \([11,12]\) of few coaches could be other than 0.

Suppose that the ‘START’ of the 3-SM S is the address 1A12h. Also suppose that the 7-train \( T_1 \) is stored in 8086 memory at the address E74Bh and the 7-train \( T_2 \) is stored at address D310h. Then the following will be incorporated in the MT:

**Upper Layer \( L_3 \) of the MT \( M \):**

The START \( M \) will point to the chief Pilot \( P = < M_1, M_2 > = < E74Bh, D310h > \).

Now, suppose that the address of the coach \( C^1_1 \) is 5008h, the address of the coach \( C^1_2 \) is A210h, and the address of the coach \( C^1_3 \) is 00AFh. Also suppose that the address of the coach \( C^2_1 \) is CA76h, the address of the coach \( C^2_2 \) is CC80h, and the address of the coach \( C^2_3 \) is BEBAh. Then the following will be incorporated in the MT:

**Middle Layer \( L_2 \) of the MT \( M \):**

It is the larray \( < T_1, T_2 > \) of two pilots \( T_1 \) and \( T_2 \) which are the two 7-Trains given by

\[
T_1 = < 5008h, A210h, 00AFh > \quad \text{and} \quad T_2 = < CA76h, CC80h, BEBAh >.
\]

From the chief pilot, one can visit directly to any of these two 7-Trains. \( M_1 \) points at the 7-Train \( T_1 \) and \( M_2 \) points at the 7-Train \( T_2 \).

**Lower Layer \( L_1 \) of the MT \( M \):**

In this layer, corresponding to each 7-Train \( T_1 \) and \( T_2 \), there 3 number of coaches, and consequently there are in total \( 3.2 = 6 \) number of coaches \( (C^1_1, C^1_2, C^1_3) \) and \( (C^2_1, C^2_2, C^2_3) \). Each coach \( C^i_j \) has 7 number of passengers, together with one more (the last one) which is an address to the next coach if \( j < 3 \),
address to the first coach of immediate higher layer if “i = 1 and j = 3”, but to an invalid address for “i = 2 and j = 3”). The status of each coach in this example is 0. From the 7-Train T₁, one can visit directly any of its coaches C₁₁, C₁₂ and C₁₃; and similarly from the 7-Train T₂ one can visit directly any of its coaches C₂₁, C₂₂ and C₂₃.

The figure-3 below shows the implementation of the data structure MT and figure-4 shows how the 3-SMS S is stored in 8086 Memory. In this example we consider a SM of small data for which the r-Trains with r = 7 have been used. However, in case r be a large number then each of these six coach to be logically divided into many sub-coaches initially, and then to be implemented by regarding each of such coaches as a r-Train or r-Atrain and the sub-coaches as the coaches. During the implementation with the data structure MT (or MA), one can use as many Trains (Atrains) as required according to the size of big data (or, according to the 4V). But for any MT or MA the lowest layer L₁ shall always consist of coaches only, not of any Train (or Atrain).

![Diagram of MT structure](image)

Figure-3. Implementation of the data structure MT for the 3-SMS S.
In this case, for storing every coach the ‘GETNODE’ will always provide sixteen number of free consecutive bytes from the memory. In each of such nodes, the first fourteen bytes contain the information and the last two bytes contain an address as explained earlier. However, $T_1$ and $T_2$ being larrays will require six bytes each. The following figure shows how this 3-SM (3-SL) S is stored in 8086 Memory starting from $START = 1A12h$ :-

<table>
<thead>
<tr>
<th>Address</th>
<th>Memory Content</th>
<th>Size</th>
</tr>
</thead>
</table>
| .......... | .......... | ......
| .......... | .......... | ......
| 0        | 2 bytes      |      |
| $C_1^3 = 00AFh$ | 2 bytes |      |
| 0        | 2 bytes      |      |
| $C_1^2 = A210h$ | 2 bytes |      |
| 0        | 2 bytes      |      |
| $E74Bh$  | $C_1^1 = 5008h$ | 2 bytes |
| .......... | .......... | ......
| .......... | .......... | ......
| 0        | 2 bytes      |      |
| $C_2^3 = BEBAh$ | 2 bytes |      |
| 0        | 2 bytes      |      |
| $C_2^2 = CC80h$ | 2 bytes |      |
| 0        | 2 bytes      |      |
| $D310h$  | $C_1^1 = CA76h$ | 2 bytes |
| .......... | .......... | ......
| BEBAh    | 2 bytes      |      |
| 0        | 2 bytes      |      |
| 3        | 2 bytes      |      |
| 9        | 2 bytes      |      |
| 7        | 2 bytes      |      |
| 1        | 2 bytes      |      |
| 8        | 2 bytes      |      |
| $CC80h$  | 3             | 2 bytes |
| .......... | .......... | ......
| $CC80h$  | 2 bytes      |      |
| 1        | 2 bytes      |      |
| 8        | 2 bytes      |      |
| 0        | 2 bytes      |      |
| 4        | 2 bytes      |      |
| 2        | 2 bytes      |      |
| 8        | 2 bytes      |      |
| $CA76h$  | 5             | 2 bytes |
| .......... | .......... | ......

Figure-4. A 3-SM (3-SL) in 8086 Memory
4.1.2 Implementation of a n-SM

For implementation of an n-SM S of size $m_1 \times m_2 \times m_3 \times \ldots \times m_n$ of homogeneous data, we need to use a MT of height n (i.e. n number of layers). In an analogous way, for implementation of a n-SM S of size $m_1 \times m_2 \times m_3 \times \ldots \times m_n$ of heterogeneous data, we need to use a MA of height n (i.e. n number of layers) as shown below.

Upper Layer $L_n$ of the MT/MA M:
Middle Layer $L_{n-1}$ of the MT/MA M:
Middle Layer $L_{n-2}$ of the MT/MA M:
.................................
.................................
Middle Layer $L_2$ of the MT/MA M:
Lower Layer $L_1$ of the MT/MA M:

Similarly, for implementing multi-dimensional arrays or multi-dimensional larrays, we can use these powerful data structures MA (if data are heterogeneous) or MT (if all data are homogeneous). In many cases, big data can be arranged as a n-SM or as a n-SL.

Consider an n-SM S of size $m_1 \times m_2 \times m_3 \times \ldots \times m_n$ of homogeneous or heterogeneous data. The size of S has n number of suffix indices $m_1$, $m_2$, $m_3$, ……, and $m_n$. If the dimension n increases dynamically with some ‘Velocity’ and/or if at least one of these n indices $m_1 \times m_2 \times m_3 \times \ldots \times m_n$ is a big integer increasing dynamically with some ‘Velocity’, then S is an ordered collection of big data in ‘Volume’ and also in other Vs out of 4Vs : Volume, Variety, Velocity and Veracity. Besides that, if S dynamically expands with respect to the parameter ‘time’ (example : a temporal big database [8,9]) or with respect to any real life parameter in at least one of the n indices, it is an example of big data in ‘Velocity’ and also in other Vs out of 4Vs. Any big data of heterogeneous datatypes can be always considered to be big in ‘Variety’ and also in other Vs out of 4Vs. The unique V ‘Veracity’ of any big data is for its completeness,
soundness, authenticity and for many other requirements or constraints which can not be compromised with any of the other Vs. Instead of big solid matrices, one could view big data in many cases in the format of multi-dimensional arrays or multi-dimensional larrays if they are dynamically expanding very fast with respect to the 4Vs: Volume, Varity, Velocity and Veracity. Any significant super V-set of the 4V-set \{Volume, Varity, Velocity and Veracity\} can add to further dimension of ‘big’ in big data. The proposed data structures MA or MT can play a great role in implementation of 4Vs while processing with big data.

In many real cases, big data can be viewed as an n-SM or an n-SL. If big data can be modeled into a SM/SL (mainly, if it is big with respect to the V ‘Volume’), the coaches of any layer in MT or MA are also loaded with big data in most of the cases. For implementation of such SM (SL) modeled big data, it is recommended that theoretically each coach to be logically divided into many sub-coaches initially, and then to be implemented by regarding each of such coaches as a Train or Atrain and the sub-coaches as the coaches. The last layer shall always consist of coaches only, not of any Trains or Atrains.

5  CONCLUSION

The work reported in this paper could be divided into three parts. In the first part, we introduce the ‘Theory of Solid Matrices’. It is expected that the notion of solid matrix (and solid latrix) will be useful mathematical objects in the mathematical modeling of problems and issues in every branch of Science, Engineering, Technology, Statistics, OR, Medical Science, etc., in particular in some cases of big data. In the second part, we introduce two powerful data structures MT and MA as useful tools to deal with big data [3,5,7,10,14], those expanding very fast in 4Vs: Volume, Varity, Velocity and Veracity. It is a matter of great concern that 4V-set has been expanding at a faster rate than 4N-set. The homogeneous data structure MT is an advancement of the homogeneous data structure Train, and the heterogeneous data structure MA is an advancement of the heterogeneous data structure Atrain [11,12]. The third part is about application
potentials. The most powerful applications of MT or MA are in the three broad directions: (i) implementation of solid matrices/latrices, (ii) implementation for multi-dimensional arrays, (iii) implementation of multi-dimensional larrays [11,12]. The various fundamental operations like: insertion, deletion, searching, sorting, etc. and the corresponding algorithms in the heterogeneous data structure Atrain (and also in the homogeneous data structure Train) are presented in [11,12]. In an analogous way, we shall extend the notion of those fundamental operations and corresponding algorithms to the case of the data structures MA and MT, but in our future work.

REFERENCES


Received: October 5, 2013