A multi-levels RNG permutation

Ammar Khaleel Abdulsadah, Abdullah Aziz Lafta, Mohammad Dosh
Department of Computer Science, Faculty of Education for Girls, University of Kufa, Iraq

Article Info

Article history:

Received Aug 25, 2019
Revised Oct 27, 2019
Accepted Nov 18, 2019

Keywords:

Algorithm
Permutation
Random-number generator
Traverse
Tree

ABSTRACT

The paper proposes a new general method for producing a multilevel permutation functioning as an m-tree traversal. It is composed of two basic steps: a random number generator of period length equal m to determine which child to traverse, and recursive permutation in which permutated the subtree if found. The test results proved that the suggested method of permutation is successful depending on the correlation measure.

Corresponding Author:

Ammar Khaleel Abdulsada,
University of Kufa,
Kufa, P.O. Box (21), Najaf Governorate, Iraq.
Email: ammar.khaleel@uokufa.edu.iq

1. INTRODUCTION

A permutation of a set of objects is a particular ordering of those objects. This supposes that permutation of the set is a one-one correspondence from this set to itself. The constraints of a permutation can be simple for a specific manner [1]. Multi-permutations and in some cases of permutations that used in various applications in an information theory [2]. permutations have a special stabilizer multiplier [3]. Random permutation is seen as one of the fundamental problems in computer science [4]. A permutation scheme used for encrypting data [5], for example, in steganography the input is decomposed into parts, then these parts are transformed into another using a permutation [6]. Also, permutation schemes proposed for group analyses [7]. Using tree data structure concept to producing the permutations is one of the methods that used for obtaining a set of permutations [8-10]. Specifically, traverse of the tree has been utilized. However, it will enable to generate one permutation at a time. There are three methods of tree traversing: preorder, inorder, and postorder [11]. The different among these methods is that the order of visiting the parent node with respect the visiting of the child nodes.

The preorder traverse is used in this work. Since, the parent node is visited before its child nodes. This allow to get the information firstly from the parent node then from child nodes. To continue tree traversing, the depth-first search approach is applied. The depth-first search accelerates the arriving to the leaves of tree without need to continue visiting the other nodes in the same level. For a moment, this lets the tree traversing to outrun many nodes on all levels (from the top of tree) to reach fast the bottom nodes. Following the depth-first search approach, when the leaves are reached, traversing will going upward to the nearest parent to visit other child nodes [12].

While each child node is associated with a distinct key value, and depending on a certain permutation for these values, then the order of visiting child nodes can be determined according to their values order on that permutation. If in some time the permutation has been changed, the traversing consequently also will be changed. Actually, the work here focuses on two issues: the key values that are kept in the nodes and the order of reach them.

The random number generator (RNG) is the subsequent generation of the integer sequence via the iterative equation [13-14]. RNG has a finite-state function that is capable of generating sequences of states which appear random-like from many aspects [15]. The RNG is a recursive relation that use an output from previous iteration as an input for the next iteration. The sequence of states has a full length when the RNG not repeat any state before all the possible states have been generated [16]. The RNG relation depends in initial state(s) [17].

In this paper, we develop a technique for permutation a sequence of values depending on the permutation of visiting child nodes when traversing through tree for suffling data. To change the order of child visiting, distinct values are given to each node to be used as a state values in a RNG. So, according to the state value generated by the RNG, the child node to be visited is determined. The recursive relation of RNG needs to recycle the same values after certain number of iterations. Suppose, for a positive integer value m, that m denotes the length of a full length sequences in RNG. An m-tree utilized these criteria.

2. PROPOSED PERMUTATION

For the problem of permutation, there are two given sets, a set X of n objects x_i and a set Y of n objects y_i , and asked to determine the permutation P:X→Y such that y_i = P(x_i) for i=1 ... n, where X and Y two sets their have exactly the same elements.

We use m-tree, where child nodes for each parent node are valued 0 to m-1. We assumed 0 as the value of the leftmost node, and m-1 as the value of the rightmost node. Figure 1 shows a diagram for m-tree.

Definition 1: An m-tree is a tree such that each non-leaf node has exactly m child nodes. In which every child node has been assigned sequentially an integer distinct value between 0 and m-1 from the leftmost node to the rightmost node.

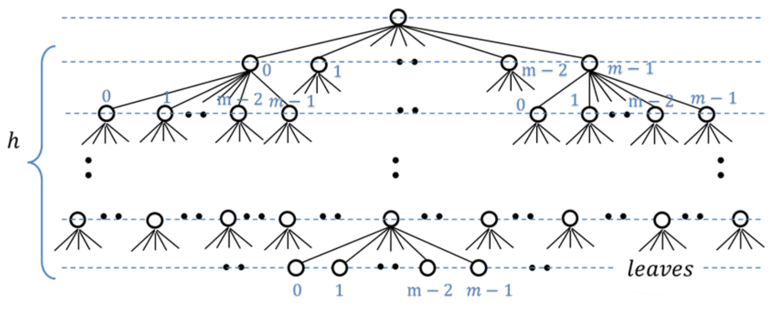

Figure 1. m-tree

In m-tree, every node has exactly m children, and all leaf nodes are at the same depth. If an m-tree has a height h, then the root node is not included in h, and it is given a height of 0. Constructing the m-tree is value-independent with the input sequence that has to be permutated. On the other hand, the length of the input sequence to be same as the number of leaves.

2.1. M-Tree and RNG

For the permutation aim, the traversing path used to reach the leaves made unfixed. Controlling the changes in path done by using the states sequence of a RNG. The initial of the used RNG is taken as an argument for next level of m-tree. The used RNG works under the assumption that no state value is repeated unless all the m values are generated.

Definition 1: The random sequence of size m, S_m, is an ordered sequence of distinct m nonnegative integer state values, $\langle r_0, \cdots, r_{m-1}\rangle$, generated by a random number generator, such that $\forall\, r_s \in S_m: 0 \le r_s < m$, and $r_s \neq r_v$ for any $s \neq v$, where $0 \le s, v < m$.

For example, if $\mathrm{m} = 16$, then 5 states values in the sequence S_{16} could be $\mathrm{r}_0 = 5$, $\mathrm{r}_1 = 14$, $\mathrm{r}_2 = 0$, $\mathrm{r}_3 = 9$, $\mathrm{r}_4 = 2$. Since, the child nodes in m-tree have only the values from 0 to $\mathrm{m} - 1$, then the generated state value can mention the child node number that has the order to be visited. This introduce what we called random m-tree.

Definition 2: A random m-tree, denoted by r-m-tree, is an m-tree, where child nodes are visited in order as the values in a sequence S_m.

That is, the value of r_0 defines the child node number that visited firstly, and the value r_1 defines the child node number that visited secondly, and so on. In r-m-tree, all nodes must be linked to one and only one value of r_s, for $0 \leq s < m$.

Definition 3: The parent node value of a child node $r_s \in S_m$ is denoted by $\pi(r_s)$.

2.2. R-M-Tree Traverse Approach

The r-m-tree levels are called permutation levels, numbered in decreasing values from root with permutation level value h, shortly written as $pl = h$, to leaves with value $pl = 0$. Hence, if r_s in $pl = j$, then $\pi(r_s)$ is in $pl = j + 1$.

Following the depth-first search principle used in graphs data representation, the traversing approach start from the root node, until reach the leaves. When reaching a node in any pl, and before go depth to a lower level, wait for one r_s value to be generated, to give a priority of visiting to a node. This principle is recurring always. Therefore, various order of sequence S_m elements, means different order in visiting nodes, consequently, the path to go ahead is different.

The main constraint is each node in r-m-tree traversing is visited only once through all the traversing process. Since, a node r_s in $pl = 1$ is not visited twice, therefore any child node x, where $r_s = \pi(x)$, is must be visited in that time, before goes to next node in $pl = 1$. The visiting order of leaves depends on state values S_m. When all leaves that belong to a certain parent node r_s in $pl = 1$ have been visited, traversing goes back to visiting another node x in $pl = 1$, which $\pi(x) = \pi(r_s)$. Also, the visiting order depends on permutated S_m. To generalize traversing approach, a triangle sub tree notion is introduced.

Definition 5. A triangle r-m-tree of r_s in $pl = j$, denoted by $\Delta^j_{r_s}$, is a sub r-m-tree whose root value $r_s \in S_m$ occurs in $pl = j$, such that $j > 0$.

Error! Reference source not found. shows a diagram for triangle of r_s in $pl = j$. For example, Δ^2_{10} denote for a r-m-tree which has a root value 10 in $pl = 2$.

Definition 4: For $j > 0$, and $r_{s_1}, r_{s_2} \in S_m$, it say a node $r_{s_2} \in \Delta^j_{r_{s_1}}$ when $r_{s_1} = \pi(r_{s_2})$.

Visually, a r-m-tree is a collection of triangle r-m-trees. Also, for any $\Delta^j_{r_s}$ and $j > 1$, there are exactly m triangle r-m-trees in the $pl = j - 1$. As a conclusion, $\Delta^j_{r_s}$ for some value r_s, represent for a randomly complete-visited tree.

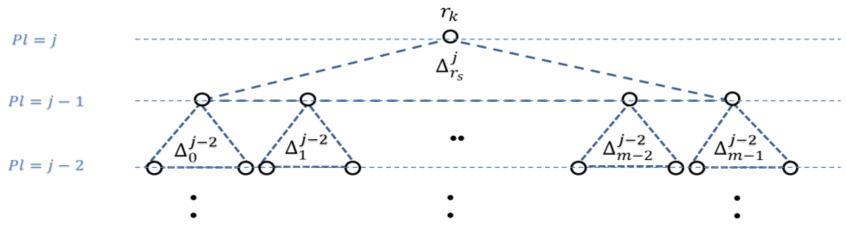

Figure 2. A triangle r-m-tree of r_s in permutation level j

Definition 5: Let $\{\Delta^j_{r_s}\}$ is the set of all leaves in $\Delta^j_{r_s}$, and $|\Delta^j_{r_s}|$ is the size of $\{\Delta^j_{r_s}\}$.

Proposition 1: For $j > 0$, $|\Delta^j_{r_s}| = m^j \ \forall\, r_s \in S_m$.

Proof. If $j > 0, \forall r_s \in S_m$ node in $pl = j$ has m child nodes, then

$$|\Delta^j_{r_s}| = \prod_{i=1}^{j} m = m^j. \blacksquare$$

Proposition 2: For any two positive integers h and j such that $j < h$, there are m^{h-j} triangle r-m-trees of all r_s in $pl = j > 0$ in a r-m-tree of depth h.

Proof. From Proposition 1, each r_s in $pl = j > 0$ has $|\Delta^j_{r_s}| = m^j$ leaf nodes, similarly, the root node of depth $h > 0$, has $|\Delta^h_{root}| = m^h$ leaf nodes, therefore the number of nodes in level j is

$$\frac{|\Delta^h_{root}|}{|\Delta^j_{r_s}|} = \frac{m^h}{m^j} = m^{h-j} \ \blacksquare$$

Definition 6: Let $S_\Delta = \{\Delta^j_{r_s}: 0 < j \leq h, r_s \in S_m\}$ called the set of all triangle r-m-trees in r-m-tree of depth $h > 1$. And $|S_\Delta|$ is size of S_Δ.

Proposition 3: If r-m-tree of depth $h > 1$ then

$$|S_\Delta| = \frac{m^h - 1}{m-1}$$

Proof. Since, the total number of triangle r-m-trees is equal the summation of triangle r-m-trees in all $pl = j > 0$ of an r-m-tree, then the solution of geometric series is in the following equation [18].

$$|S_\Delta| = \sum_{j=1}^{h} m^{h-j} = \frac{m^h - 1}{m-1} \blacksquare$$

2.3. Offset Calculation

Each node in r-m-tree assigned two values; a relative offset and an absolute offset. These two values are depend on the value of m and the height of tree h . The traverse path to reach a leaf node, measured by an absolute offset of its parent.

Definition 7: The relative offset value of $\Delta^j_{r_s}$, denoted by $\partial(\Delta^j_{r_s})$, is the sum of $|\Delta^j_x|$ for any $0 \leq x < r_s$, such that $\pi(r_s) = \pi(x)$.

Proposition 4: $\partial(\Delta^j_{r_s}) = r_s \cdot m^j$.

Proof. From the Definition 7, and use Proposition 1, if $0 \leq x < r_s$ then $\partial(\Delta^j_{r_s})$, can be written as,

$$\partial(\Delta^j_{r_s}) = \sum_{x=0}^{r_s-1} |\Delta^j_x| = \sum_{x=0}^{r_s-1} m^j = r_s \cdot m^j \blacksquare$$

Definition 8: The absolute offset value of $\Delta^j_{r_s}$, denoted by $\varphi(\Delta^j_{r_s})$, is the sum of $\partial(\Delta^i_{x_i})$ for any $j \leq i < h$, such that $x_j = r_s$ and $x_{i+1} = \pi(x_i)$.

As a result of proposition 4, the absolute offset value of $\Delta^j_{r_s}$ be computed as,

$$\varphi(\Delta^j_{r_s}) = \sum_{i=j}^{h-1} \partial(\Delta^i_{x_i}) = \sum_{i=j}^{h-1} (x_i \cdot m^i)$$

2.4 r-m-Tree Permutation Method

Let an input sequence $X = \langle x_0 . x_1 . \cdots . x_{N-1} \rangle$, where N is equal to $|\Delta^h_{root}|$ in r-m-tree of depth h. Also, suppose an output sequence $Y = \emptyset$ in the beginning of algorithm. The following formula describes the permutation decision:

$$Y = Y + x_{r_s + \varphi(\Delta^1_{\pi(r_s)})} \; \forall \, r_s \in \Delta^1_{\pi(r_s)} \quad (1)$$

That is, in the current-end position of Y, add the element of X whose index is equal to the result of adding the value of r_s to the absolute offset value $\varphi(\Delta^1_{\pi(r_s)})$. Set Y works as a queue to register the order of adding the elements, which is the reason for the permutation to be done this way. Formula (1) is generalized by Triangle Permutation, TP, algorithm to any offset could be calculated. The notation Δ^1_x, for any node value of x, represent a randomly complete-visited sub tree in $pl = 1$, that has m children.

For permutation purposes, must repeat the formula (1) for all $r_s \in \Delta^1_{\pi(r_s)}$. If $r_{s_2} = \pi(r_{s_1})$, then the formula (1) is repeated with all triangles in $pl = 1$, according to the following loop:

For each $r_{s_2} \in \Delta^2_{\pi(r_{s_2})}$ do

$$Y = Y + x_{r_{s_1} + \varphi(\Delta^1_{r_{s_2}})} \; \forall \, r_{s_1} \in \Delta^1_{r_{s_2}} \quad (2)$$

The proposed Multi-Level Permutation, MLP, algorithm works recursively to generalize the formula (2) to cover all triangles in r-m-tree, under assumption that a session of triangle to be permuted is launched by the traverse path, and no triangle permuted twice. **Error! Reference source not found.** shows the proposed algorithms. **Error! Reference source not found.** shows a diagram for possible relation between sequences X and Y.

```
Input: An offset $\varphi\left(\Delta_{r_{s_1}}^{j}\right)$.
        permutation level $j$. and a sequence $X$ (a global variable).
Output: A sequence $Y$ (a global variable).
Algorithm $MLP\left(\varphi\left(\Delta_{r_{s_1}}^{j}\right), j\right)$
    if $j = 1$ then
        $TP\left(\varphi\left(\Delta_{r_{s_1}}^{j}\right)\right)$
    else
        for each $r_{s_2} \in \Delta_{r_{s_1}}^{j}$ do
            $MLP\left(\partial\left(\Delta_{r_{s_2}}^{j-1}\right) + \varphi\left(\Delta_{r_{s_1}}^{j}\right), j-1\right)$
Algorithm $TP\left(\varphi\left(\Delta_{r_{s_1}}^{1}\right)\right)$
    for each $r_{s_2} \in \Delta_{r_{s_1}}^{1}$ do
        $Y = Y + x_{\varphi\left(\Delta_{r_{s_1}}^{1}\right)+r_{s_2}}$
```

Figure 3. TP and MLP algorithms

Because the traverse is starting from the root node of a tree, which has an absolute offset value 0 and pl = h, then the first call to start use the proposed algorithm is MLP(0. h). As with normal algorithm recursion calls, a stack records all previous values were used before the call. MLP algorithm passes a new value for absolute offset each time there is a call to a deeper level. When the call ends, it reuse the previous absolute offset.

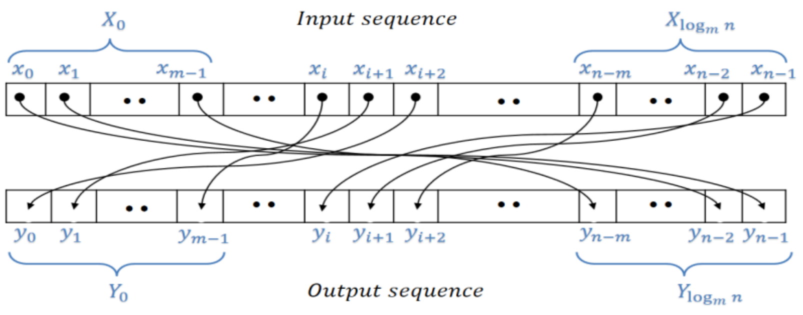

Figure 4. Possible relation between input and output sequences

Proposition 5: MLP algorithm is correct.

Proof. MLP algorithm is an application of depth-first traverse under preorder approach, only the difference is the order of visiting child nodes that done according to the sates generated by a used RNG. Since the permutation is states dependence, the correctness of RNG states is ensure the correctness of proposed algorithm.

Proposition 6: If the input sequence has m^h elements, $h > 0$, then the running time of MLP algorithm is $\Theta(m^h)$.

Proof. The recurrence relation for MLP algorithm is the following:

$$T(m^h) = \begin{cases} \Theta(m) & if\ h = 1 \\ m \cdot T(m^{h-1}) + \Theta(1) & if h > 1 \end{cases}$$

Solving the recurrence give us: $T(m^h) = \Theta(m^h)$.

3. NUMBER OF PERMUTATIONS

There is N! possible permutations for a set of N elements [19]. The proposed algorithm coordinates the permutation for a set of $N = m^h$ elements by traversing all the triangle r-m-trees. Depending on parameters of the used RNG, the MLP algorithm generates only one of these N! possible permutations in every time it is used. With successive varying in RNG parameters, the maximum number of possible permutations is,

$$(m!)^{|S_\Delta|}$$

Proposition 7: If using MLP, the maximum possible permutation for a sequence length $N = m^h$ is,

$$(m!)^{\frac{m^h - 1}{m-1}}$$

Proof. Since S_m is a sequence of m elements, there are m! possible permutation for its elements. Hence, each triangle r-m-tree has m! possible different order of visiting its m child nodes. And, because there are $|S_\Delta|$ triangle trees, then the total possible permutations for all triangle trees is $(m!)^{|S_\Delta|}$. According to Proposition 3, the total possible permutations by r-m-tree permutations is:

$$(m!)^{\frac{m^h - 1}{m-1}} \blacksquare$$

4. REVERSE THE PERMUTATION

The MLP algorithm used to reverse the permutation, i.e. get X from Y, the MLP algorithm is used to call Inverse Triangle Permutation, TP^{-1}, algorithm when the $pl = 1$. **Error! Reference source not found.** shows the steps of TP^{-1} algorithm. The absolute offset $\varphi\left(\Delta^1_{r_{s_1}}\right)$ is added to the value of r_{s_2} to calculate the index of location in X where to put the value of current-first element of Y, denoted by y_0. The set Y is used as queue where y_0 is the extracted element from the beginning of queue Y. After each allocation step, a new element y_0 is taken from Y.

```
**Algorithm** $TP^{-1}\left(\varphi\left(\Delta^1_{r_{s_1}}\right)\right)$
    $for\ each\ r_{s_2} \in \Delta^1_{r_{s_1}}\ do$
        $x_{\varphi\left(\Delta^1_{r_{s_1}}\right)+r_{s_2}} = y_0$
        $Y = Y - y_0$
```

Figure 5. Inverse triangle permutation

Proposition 8: Algorithm TP^{-1} is correct.

Proof. MLP algorithm calculate $\varphi\left(\Delta^1_{r_{s_1}}\right)$ value, and TP algorithm generate r_{s_2} value, then the element of X at location $(\varphi\left(\Delta^1_{r_{s_1}}\right) + r_{s_2})$ is pushed at the end of Y. That is, the elements of Y are in the same order they were added. MLP algorithm is used again to calculate the same $\varphi\left(\Delta^1_{r_{s_1}}\right)$ value, and TP^{-1} algorithm generate r_{s_2} in the same way TP algorithm does, then the element extracted from the beginning of Y into the location $(\varphi\left(\Delta^1_{r_{s_1}}\right) + r_{s_2})$ of X. $\blacksquare$

The running time for inverse permutation is also $\Theta(m^h)$.

5. TESTING AND PRACTICAL WORKS

In testing MLP algorithm, linear congruential generator (LCG) is used because of simplicity as a RNG [20]. LCG is a recurrence relation yields a sequence of nonnegative numbers defined as:

$$S_{i+1} = (\ a \cdot S_i + b\ ) \bmod m$$

For all levels, the following LCG parameters are used:
$m = 64, a = 189, b = 47, S_0 = 50$

The correlation coefficients between the original sequence and the permutated sequence for each sample are shown in **Error! Reference source not found.**. The samples are selected according to differences in their original histograms. The correlation results for 24 sample set for Xs, where each sequence $X = \{x: 0 \leq x \leq 255\}$, and $|X| = 65536$ elements. Each sequence X is divided to 16 parts. Each part of $(64)^2$ elements is implemented as r-64-tree of depth $h = 2$ visited according to LCG random generator. The correlation is calculated for the sequence X as one set.

Table 1. Sample of Correlation Results

X#	Correlation	X#	Correlation	X#	Correlation
1	0.072321126	9	0.018605276	17	0.00620841
2	0.47566501	10	0.013133362	18	0.012102133
3	0.100649146	11	0.06639886	19	0.027639506
4	0.061073278	12	0.025166674	20	0.010722534
5	0.015872294	13	0.078569469	21	0.027580224
6	0.023976741	14	0.07015351	22	0.01103299
7	0.01597167	15	0.090477404	23	0.017008948
8	0.010068363	16	0.068917077	24	0.099960463

6. RESULTS

An r-m-tree works as, firstly, divide an input sequence of N elements into m partitions, secondly, each partition is also divided into m sub partitions, and so on, until reach a partition of only one element inside. A permutation decision is taken, when reaching a leaf node, and all other crossed nodes through the traversing path are used for stretch the permutation. Generally, increasing the number of permutation levels of the tree will engage more possible paths.

We find that our proposed algorithm is a permutation to m-block objects, where each object itself could be a permutation to m-block objects or its simple a permutation to m elements. This can be seen clearly if the height of tree is 1, in this time $N = m$, which is become a permutation for one block of size N. In this situation, the quality of permutation is full-dependent on the used random number generator for our proposed MLP algorithm. The proposed permutation algorithm is used for permutation or inverse permutation without slightly change in two algorithms: TP and TP^{-1}.

7. CONCLUSION

In this paper, we have addressed a new method for producing a multilevel permutation functioning. The proposed method depends mainly on two basic steps, random number generator (RNG) to determine which child to traverse, and recursive permutation in which permutated the subtree. Our algorithm takes only $O(n)$ time to getting one permutation out of $(m!)^{\frac{m^h-1}{m-1}}$ possible permutations. We develop a continuation-based method for finding a permutation using the new parametrization and the RNG.

The permutation is a changeable because of its generating method that is depending completely on a RNG that be selected in implementation.

REFERENCES

[1] C. H. Lim and S. Wright, "*A Box-Constrained Approach for Hard Permutation Problems,*" Proc. 33rd Int. Conf. Mach. Learn., vol. 48, pp. 2454–2463, 2016.

[2] S. Buzaglo, E. Yaakobi, T. Etzion, and J. Bruck, "Systematic Error-Correcting Codes for Permutations and Multi-Permutations," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3113–3124, 2016.

[3] I. A. Joundan, S. Nouh, M. Azouazi, and A. Namir, "A new efficient way based on special stabilizer multiplier permutations to attack the hardness of the minimum weight search problem for large BCH codes," vol. 9, no. 2, pp. 1232–1239, 2019.

[4] Z. Palmer and J. Riley, "An Empirical Analysis of Parallel Online Learning Algorithms," Methodology, 2006.

[5] W. Kang and N. Liu, "Compressing Encrypted Data and Permutation Cipher," no. 2014, pp. 1–17, 2014.

[6] Y. Taouil and E. B. Ameur, "Steganographic Scheme Based on Message-Cover matching," 2018.

[7] J. A. Etzel, "MVPA Permutation Schemes: Permutation Testing for the Group Level," Proc. - 2015 Int. Work. Pattern Recognit. NeuroImaging, PRNI 2015, pp. 65–68, 2015.

[8] B. Cheng, J. Fan, and X. Jia, "Dimensional-permutation-based independent spanning trees in bijective connection networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 1, pp. 45–53, 2015.

[9] S. Jiang, F. Mo, F. C. M. Lau, and C. W. Sham, "Tree-Permutation-Matrix Based LDPC Codes," *IEEE Trans.*

Circuits Syst. II Express Briefs, vol. 65, no. 8, pp. 1019–1023, 2018.
[10] Uthan, Seeniya, and B Chitturi, "*Bounding the diameter of cayley graphs generated by specific transposition trees.*" 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, 2017.
[11] M. K. Saparbaev, a V Mazin, L. P. Ovchinnikova, G. L. Dianov, and R. I. Salganik, Cormen - Projeto e Análise de Algoritmos, no. 2. 1988.
[12] S. Khan, "Depth First Search in the Semi-streaming Model," no. 340506, pp. 1–25, 2013.
[13] M. O. Steinhauser, E. H. Sibley, S. K. Park, and K. W. Miller, "RANDOM NUMBER GEUERATORS : GOOD ONES ARE HARD TO FIN," no. 10, 2014.
[14] N. Alia, N. Hashim, J. Teo, H. Loong, A. Ghazali, and F. A. Hamid, "Memristor based ring oscillators true random number generator with different window functions for applications in cryptography," vol. 14, no. 1, pp. 201–209, 2019.
[15] T. Stojanovski, "Chaos-Based Random Number Generators — Part I : Analysis," no. 2014, 2001.
[16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "*Numerical Recipes in C The Art of Scientific Computing.*"
[17] A. A. Rezk, A. H. Madian, A. G. Radwan, and A. M. Soliman, "Reconfigurable Chaotic Pseudo Random Number Generator based on FPGA," *AEUE - Int. J. Electron. Commun.*, 2018.
[18] Flanders, Harley, and J J. Price, "Calculus with analytic geometry," Academic Press, 2014.
[19] KH Rosen,and K Krithivasan, "QUADRATICS (Discrete Mathematics and its Applications)," Tata McGraw-Hill Education, 2012.
[20] K. Hongo, R. Y. O. Maezono, and K. Miura, "Random Number Generators Tested on Quantum Monte Carlo Simulations," 2010.

BIOGRAPHIES OF AUTHORS

Ammar Khaleel Abdulsada is an Assistant Lecturer of Computer Science an administrator of E-learning department at the University of Kufa. Obtained his a bachelor's degree in Computer Science from Mustansiriyah University in Iraq and master degree from the College of Computing & Informatics, Universiti Tenaga Nasional (UNITEN) in Malaysia. His research interests include Networking and Internet of Things (IoT.

Abdullah Aziz Lafta recieved his Bachelor of Computer Science from the Faculty of Science in Kuwait University, Kuwait in 1988 and Master of Computer Science from the Faculty of Science in AL-Nahrain University, Iraq 1995.He joined the University of Kufa in 2011. He worked in educational institutions in several Arab countries. Main research interests are Image processing and computer security.

Mohammed Dosh is a lecturer in Computer department in University of Kufa. He received his master degree in Computer Sience form SHIATS University, India, and his Phd in Computer Applied Technology from Huazhong University of Science & Technology, China. His research interests Artificial Intelligence, Data Mining, Computer Applied Technology, Design Algorithms, and Natural Language Processing.