IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. VIII (Feb. 2014), PP 76-80
www.iosrjournals.org

Modified Method of Generating Randomized Latin Squares

D. Selvi[1] G. Velammal[2] ThevasahayamArockiadoss[3]
[1]*Research Associate, Oxysynth India Private. Ltd., Madurai, India*
[2]*Department of Mathematics, Sri Meenakshi Government College (W) Madurai, India*
[3]*Department of Physics, Madurai Kamaraj University-DDE, Madurai, India*

Abstract: *In combinatorics and in experimental design, Latin square (LS) plays a vital role. It has been studied for centuries and especially used in statistics. Let L(n) be the number of Latin square of order n . L(n) increases exponentially as n increases. The feasible solutions of three dimensional planar assignment problem (3DPAP) are latin squares. The 3DPAP and related problems are NP hard. Hence to solve these problems using genetic algorithms, randomly creating a Latin square method is essential. O'Carroll has given a method for random generation of Latin square. The method however fails occasionally. So we have given a modified method and this modified method is efficient in the generation of random Latin square.*
Keywords: *Latin square, Random*

I. INTRODUCTION

Generating random Latin square(LS) is a relevant topic. Jacobson and Matthews[1] have described an algorithm based on moves for generating one LS from another LS, and the moves are such that there is a way to get from each square to every other square. R.Fontana [2] gave method based on equivalence between Latin squares and Maximum cliques of a graph. Czeslaw [3] has proposed a method of generating Quasigroups for cryptography application, since Latin square is an essential element for the implementation of both a key generator and the enciphering / deciphering procedure,random generation of Latin square is useful for cryptography application. The feasible solution of the planar three dimensional assignment problems is Latin square. The planar three dimensional assignment problem has been studied by many mathematicians. M.Frieze[4] has showed that the planar three index assignment problem is NP-hard. It has several applications with respect to scheduling and timetabling problems. Hence the random generation method for Latin square is important for many real world applications. O'Carroll [5] has given method of generating randomized Latin square, this method failed for minimal assigning identical set. Hence in this paper we present a modified method to O.Carroll's randomized generation of Latin square method. This approach is very useful to generate Latin square of any order.

II. LATIN SQUARE

A Latin square is an n × n array filled with n different Latin letters, each occurring exactly once in each row and exactly once in each column. For any positive integer n there exists a Latin square of order n.

2.1 THE NUMBERS OF LATIN SQUARES OF VARIOUS SIZES

A Latin square is said to be reduced (also, normalized or in standard form) if both its first row and its first column are in their natural order. L (n) = (n-1)! n! R (n),where R (n) is the number of reduced Latin square of order n. The following table-1 gives us some idea about L (n) and R (n).

2.2 EQUIVALENCE CLASSES OF LATIN SQUARES

The following operations change a Latin square to another Latin square of same order:

(i) Interchanging of rows
(ii) Interchanging of columns
(iii) Let $1 \le p < q \le n$. Changing every p in the LS into q and every q to p.
(iv) Transposing the Latin square.
(v) Consider LS with entry k in i th row j th column. Construct a square with entry i in k th row, j th column .That square too will be a LS.
(vi) Similarly for every LS with entry k in i th row j th column , the square constructed with entry j in the i th row k th column too will be a LS.

When we get a Latin square B by performing series of such operations on a Latin square A, let us write ARB, where R is a relation. When ARB, A and B are said to be Para topic. It can be verified that R is an equivalence relation. Hence R will partition the L (n) Latin squares of order n into equivalence classes called

main classes. If we permute the rows, permute the columns, and permute the names of the symbols of a Latin square, we obtain a new Latin square said to be, so the set of all Latin squares is divided into subsets, called isotropy classes, such that two squares in the same class are isotopic and two squares in different classes are not isotopic.

TABLE 1

The numbers of Latin squares of various orders

N	R(n)	L(n)
1	1	1
2	1	2
3	1	12
4	4	576
5	56	161280
6	9408	812851200
7	16942080	61479419904000
8	535281401856	108776032459082956800
9	377597570964258816	5524751496156892842531 22
10	7580721483160132811 48	9982437658213039871725 06475692
11	5363937773277371298 119673540771840	7769668361717701441074 44346734230682311065600000

TABLE 2

Equivalence classes of Latin square

	Main classes	Isotropy classes
1	1	1
2	1	1
3	1	1
4	2	2
5	2	2
6	**12**	**22**
7	**147**	**564**
8	**283657**	**1676267**
9	**19270853541**	**115618721533**
10	**34817397894749939**	**208904371354363006**
11	**203602955252883134196099**	**12216177315369229261482540**

Hence, as n increases L (n) increases exponentially.

We note that the number of Latin squares of order 10 is more than 10^{37} and that the number of main classes in this case is more than 34 quadrillion. Hence there is essential to create a Latin square randomly.

III. O' Carroll's Method To Generate Random Latin Square

The square is constructed row by row, but within each row the elements are not necessarily entered in regular order. Consider the position when, in constructing an N x N Latin square, the first R - 1 rows have been completed and in the R^{th} row I - 1 elements have been filled in. Let A_J be the number of different letters that can be inserted in the J^{th} position in the R^{th} row without violating the condition for a Latin square. If the J^{th} position has already been filled then $A_J = 0$, otherwise J is obtained by counting the number of different letters that have not already been included in either the R^{th} row or the J^{th} column. Let A_{N+K} be the number of different positions in the R^{th} row in which the K^{th} letter of the alphabet can be inserted. If the K^{th} letter of the alphabet has already been used in this row then $A_{N+K} = 0$. Let A_S be the smallest non-zero value in the set $A_1, A_2, ..., A_{2N}$, and B be a random integer in the range 1 to A_S. (If there is a tie, the choice of A_S among the subset of jointly smallest elements is immaterial; that with the smallest value of S within this subset can conveniently be taken). The next step in constructing the square is then determined by the values of S and B, as follows",

"(i) If S <= N, insert in the S^{th} position in the R^{th} row the B^{th} letter among those that can be entered in this position."

"(ii) If S > N, insert the $(S - N)^{th}$ letter of the alphabet in the B^{th} position among those still open to it in the R^{th} row."

The above method works well most of the times. When equal sets come, this method fails. We give an example: While generating a 9×9 Latin squares using the above method, the first 6 rows were completed. The problem occurred when the 7^{th} row was being filled

TABLE 3

Sample for O' Carroll's method's failure to generate Latin square

I-Row	6	9	5	2	1	3	8	4	7
II-Row	9	5	8	1	3	4	6	7	2
III-Row	2	4	7	9	8	6	1	5	3
IV-Row	7	2	1	3	9	8	5	6	4
V-Row	1	8	3	7	5	2	4	9	6
V1-Row	4	3	9	6	7	5	2	8	1
VII-Row	3(step-5)	6(step-4)	4(step-8)	5(step-6)	?	1(step-3)	7(step-2)	2(step-7)	9 (step-1)

The following table gives explanation of O Carroll's method failed to generate the 7 th row of the above table.

Position	Elements to be filled in the corresponding Position	Number of elements	**Step-1** (suppose in 9th position element 9 is filled)	Position	Elements to be filled	Number of elements	**Step-2** (in 7 th position element 7 is filled	Position	Elements to be filled	Number of elements	**Step-3** (6 th position is selected and element 1 is filed
A1	[3 5 8]	3	3	A1	[3 5 8]	3	3	A1	[3 5 8]	3	3
A2	[1 6 7]	3	3	A2	[1 6 ~~7~~]	3	2	A2	[~~1~~ 6 ~~7~~]	2	1
A3	[2 4 6]	3	3	A3	[2 4 6]	3	3	A3	[2 4 6]	3	3
A4	[4 5 8]	3	3	A4	[4 5 8]	3	3	A4	[4 5 8]	3	3
A5	[2 4 6]	3	3	A5	[2 4 6]	3	3	A5	[2 4 6]	3	3
A6	[1 7 ~~9~~]	3	2	A6	[1 ~~7~~ ~~9~~]	2	1	A6	[~~1 7 9~~]	1	0
A7	[3 7 ~~9~~]	3	2	A7	[~~3 7 9~~]	2	0	A7	[~~3 7 9~~]	0	0
A8	[1 2 3]	3	3	A8	[1 2 3]	3	3	A8	[~~1~~ 2 3]	3	2
A9	[~~5 8 9~~]	3	0	A9	[~~5 8 9~~]	0	0	A9	[~~5 8 9~~]	0	0

Element (S-N)	Elements to be filled in the corresponding Position	Number of elements	**Step-1** (suppose in 9th position element 9 is filled)	Element (S-N)	Elements to be filled	Number of elements	**Step-2** (in 7 th position element 7 is filled	Element (S-N)	Elements to be filled	Number of elements	**Step-3** (6 th position is selected and element 1 is filed
A10	[2 6 8]	3	3	A10	[2 6 8]	3	3	A10	[~~2 6 8~~]	3	0
A11	[3 5 8]	3	3	A11	[3 5 8]	3	3	A11	[3 5 8]	3	3
A12	[1 7 8]	3	3	A12	[1 ~~7~~ 8]	3	2	A12	[1 ~~7~~ 8]	2	2
A13	[3 4 5]	3	3	A13	[3 4 5]	3	3	A13	[3 4 5]	3	3
A14	[1 4 ~~9~~]	3	2	A14	[1 4 ~~9~~]	2	2	A14	[1 4 ~~9~~]	2	2
A15	[2 3 5]	3	3	A15	[2 3 5]	3	3	A15	[2 3 5]	3	3
A16	[2 6 7]	3	3	A16	[~~2 6 7~~]	3	0	A16	[~~2 6 7~~]	0	0
A17	[1 4 ~~9~~]	3	2	A17	[1 4 ~~9~~]	2	2	A17	[1 4 ~~9~~]	2	2
A18	[~~6 7 9~~]	3	0	A18	[~~6 7 9~~]	0	0	A18	[~~6 7 9~~]	0	0

Position As	Elements to be filled in the corresponding Position S	Number of elements	**Step4** (2nd position is selected and element 6 is filled	Position	Elements to be filled	Number of elements	**Step5** (1st position is selected and element 3 is filled	Position	Elements to be filled	Number of elements	**Step6** (4th position is selected and element 5 is filled
A1	[3 5 8]	3	3	A1	[3 5 8]	3	0	A1	[3 5 8]	0	0
A2	[1 6 7]	0	0	A2	[1 6 7]	0	0	A2	[1 6 7]	0	0
A3	[2 4 6]	0	0	A3	[2 4 6]	2	2	A3	[2 4 6]	2	2
A4	[4 5 8]	3	3	A4	[4 5 8]	3	3	A4	[4 5 8]	3	0
A5	[2 4 6]	2	2	A5	[2 4 6]	2	2	A5	[2 4 6]	2	2
A6	[1 7 9]	0	0	A6	[1 7 9]	0	0	A6	[1 7 9]	0	0
A7	[3 7 9]	2	0	A7	[3 7 9]	0	0	A7	[3 7 9]	0	0
A8	[1 2 3]	2	2	A8	[1 2 3]	2	1	A8	[1 2 3]	1	1
A9	[5 8 9]	3	0	A9	[5 8 9]	0	0	A9	[5 8 9]	0	0

Element (S-N)	Position to be filled	Number of elements	Element 6 is filled in 2nd position	Element (S-N)	Position to be filled	Number of positions	Element 3 is filled in 1st position	Element (S-N)	Position to be filled	Number of positions	Element 5 is filled in 4th position
A10	[2 6 8]	0	0	A10	[2 6 8]	0	0	A10	[2 6 8]	0	0
A11	[3 5 8]	3	3	A11	[3 5 8]	3	3	A11	[3 5 8]	3	3
A12	[1 7 8]	2	2	A12	[1 7 8]	2	0	A12	[1 7 8]	0	0
A13	[3 4 5]	3	3	A13	[3 4 5]	3	3	A13	[3 4 5]	3	2
A14	[1 4 9]	2	2	A14	[1 4 9]	2	1	A14	[1 4 9]	1	0
A15	[2 3 5]	3	0	A15	[2 3 5]	0	0	A15	[2 3 5]	0	0
A16	[2 6 7]	0	0	A16	[2 6 7]	0	0	A16	[2 6 7]	0	0
A17	[1 4 9]	2	2	A17	[1 4 9]	2	1	A17	[1 4 9]	1	0
A18	[6 7 9]	0	0	A18	[6 7 9]	0	0	A18	[6 7 9]	0	0

Position	Elements to be filled in the corresponding Position	Number of elements	**Step-7** (in 8th position element 2 is filled))	Position	Elements to be filled	Number of elements	**Step-8** (in 3rd position element 4 is filled)	Position	Elements to be filled	Number of elements	**Step-9 (There is no non zero "A_s" to allocate)**
A1	[3 5 8]	0	0	A1	[3 5 8]	0	0	A1	[3 5 8]	0	
A2	[1 6 7]	0	0	A2	[1 6 7]	0	0	A2	[1 6 7]	0	
A3	[2 4 6]	2	1	A3	[2 4 6]	1	0	A3	[2 4 6]	0	
A4	[4 5 8]	3	0	A4	[4 5 8]	0	0	A4	[4 5 8]	0	
A5	[2 4 6]	2	1	A5	[2 4 6]	1	0	A5	[2 4 6]	0	
A6	[1 7 9]	0	0	A6	[1 7 9]	0	0	A6	[1 7 9]	0	
A7	[3 7 9]	0	0	A7	[3 7 9]	0	0	A7	[3 7 9]	0	
A8	[1 2 3]	1	0	A8	[1 2 3]	0	0	A8	[1 2 3]	0	
A9	[5 8 9]	0	0	A9	[5 8 9]	0	0	A9	[5 8 9]	0	

Element (S-N)	Position to be filled	Number of elements	Element 2 is filled in 8th position	Element (S-N)	Position to be filled	Number of elements	Element 2 is filled in 8th position	Element (S-N)	Position to be filled	Number of elements	
A10	[2 6 8]	0	0	A10	[2 6 8]	0	0	A10	[2 6 8]	0	
A11	[3 5 8]	3	0	A11	[3 5 8]	0	0	A11	[3 5 8]	0	
A12	[1 7 8]	0	0	A12	[1 7 8]	0	0	A12	[1 7 8]	0	
A13	[3 4 5]	2	2	A13	[3 4 5]	2	0	A13	[3 4 5]	0	
A14	[1 4 9]	0	0	A14	[1 4 9]	0	0	A14	[1 4 9]	0	
A15	[2 3 5]	0	0	A15	[2 3 5]	0	0	A15	[2 3 5]	0	
A16	[2 6 7]	0	0	A16	[2 6 7]	0	0	A16	[2 6 7]	0	
A17	[1 4 9]	0	0	A17	[1 4 9]	0	0	A17	[1 4 9]	0	
A18	[6 7 9]	0	0	A18	[6 7 9]	0	0	A18	[6 7 9]	0	

In Step-9 there is no non zero "A_s" to allocate in 5th position. Hence O' Carroll's method has failed in this case

IV. MODIFIED O' CARROLL.F. METHOD TO GENERATE NXN LATIN SQUARE

O' Carroll stated that, if there is a tie, in the choice of A_S among the subset of jointly smallest elements is immaterial. Due to this reason the problem arise in the generation **of Latin** square. Hence we modify the method as follows:

If a particular choice leads to failure, backtrack and try another choice. In any case we do not have to backtrack to the previous row. In the worst case it is enough to start filling the current row freshly.

Using Modified O' Carroll's method, for the above example , if we first select 3rd or 5th position we can complete the 7th row of the sample Latin square table, since 3rd and 5th position have same set of elements to be filled and proceeding the same way, we get one of the following 9x9 Latin squares.

TABLE 4

Latin square generated by using Modified O' Carroll.F.

I-ROW	6	9	5	2	1	3	8	4	7
II-ROW	9	5	8	1	3	4	6	7	2
III-ROW	2	4	7	9	8	6	1	5	3
IV-ROW	7	2	1	3	9	8	5	6	4
V-ROW	1	8	3	7	5	2	4	9	6
V1-ROW	4	3	9	6	7	5	2	8	1
VII-ROW	3	1	6	8	4	7	9	2	5
VIII-ROW	8	6	4	5	2	9	3	1	9
IX-ROW	5	7	2	4	6	1	7	3	8

Hence our modified method for O'Carroll's is useful to generate random Latin Square for any order.

V. Conclusion

This paper has presented a method to generate random Latin square. The problem which occurred in the random generation method in the selection of elements or position to be filled in random generation as given by O'Carroll was studied carefully and has been rectified by modifying the method . Using the proposed modified method, we successfully generate random Latin square for any order.

Future work is focused to solve three dimensional planar assignment and related problems using Genetic algorithm by generating initial solutions using this random generation method. Further Jacobson and Mathews ± moves method has been used as a mutation method to search the entire search space by jump out of one main class to another main class to generate solutions from different parts of entire search space of set of all feasible solutions.

References

[1]. M. Jacobson, P. Matthews, Generating Uniformly Distributed Random Latin Squares, Journal of Combinatorial Designs, Vol. 4, No. 6 ,1996
[2]. Roberto Fontana ,Random Latin squares and Sudoku designs generation, IOP, air Xiv:1305.3697v1[stat.co], May 2013
[3]. Czeslaw Koscielny, Generating Quasigroups for Cryptographic Applications, International Journal of Applied Mathematics and Computer Science, 2002, vol.12.No.4 559-569
[4]. A.Frieze, complexity of a 3-Dimensional Assignment Problem, European Journal of Operational Research, 1983,**13:**161–164
[5]. O'Carroll, F, A Method of Generating Randomized Latin Squares, Biometrics. December 1963, 652-653