

LATIN SQUARE COMPLETION

PROJECT REPORT

Prit J. Kanadiya (211070010)

Vedant R. Nimje (211070015)

ABSTRACT

A Latin square of order n is an array of n symbols (i.e., $\{1, 2, \dots, n\}$) in which each symbol occurs exactly once in each row and exactly once in each column. If some grids are empty, then the Latin square complete (LSC) problem of order n aims to complete the empty grids with n symbols to obtain an arbitrary legal Latin square.

0			
	1		
		1	2
		3	1

0	3	2	1
2	1	0	3
3	0	1	2
1	2	3	0

NP-Complete Problem

Latin Square Completion is a Constraint Satisfaction Problem and hence is NP Complete in nature. This is because it can be solved only in non-deterministic polynomial time, but easy to verify in linear time ($O(n)$) time complexity.

It can be defined as a Constraint Satisfaction Problem (CSP) as follows:

Variables: Each cell in $N \times N$ matrix

Domains: $\{1, 2, \dots, n\}$

Constraints: Each symbol (1- n) can occur exactly once in a row and column

In this project, we aim to explore various state space reduction techniques and heuristics which can be used to solve this LSC problem, using graph-coloring representation, to reduce the computational complexity of the problem.

PROBLEM REPRESENTATION

An arbitrary legal Latin square L_n is an $n \times n$ array filled with n different symbols, each occurring exactly once in each row and exactly once in each column. If some grids are empty, then L_n^p is called a partial Latin square. The Latin square completion (LSC) problem aims to fill symbols (i.e., $\{1, 2, \dots, n\}$) to empty grids of L_n^p to obtain an arbitrary legal Latin square.

Jin and Hao (2019) defined a Latin square graph $G = (V, E)$ to intuitively show a partial Latin square L_n^p , where $V = \{v_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq n\}$ represents all grids and vertex v_{ij} denotes a grid on the i -th row and the j -th column.

- If two grids $u, w \in V$ are in the same row or column, then we say $(u, w) \in E$. Thus, $|V| = n^2$ and $|E| = n^2(n - 1)$.
- For each vertex $v \in V$, the neighbors of vertex v is defined as

$$N(v) = \{u \in V \mid (v, u) \in E\}$$

and the degree of vertex is $d(v) = |N(v)| = 2(n - 1)$.

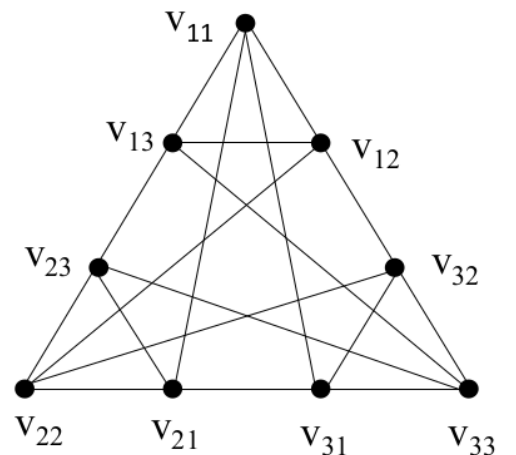
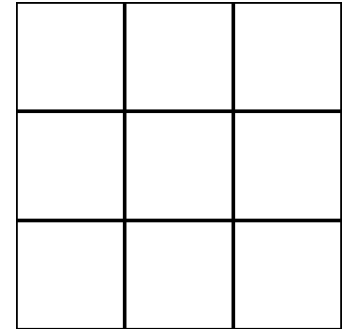
- For $\forall v_{ij} \in V$, its row vertex set is denoted as

$$RN(v_{ij}) = \{v_{ik} \mid 1 \leq k \leq n, k \neq j\},$$

while its column vertex set is denoted as

$$CN(v_{ij}) = \{v_{kj} \mid 1 \leq k \leq n, k \neq i\}.$$

Hence, $N(v) = RN(v) \cup CN(v)$.



A legal n -coloring is a partition of V into n independent sets (color classes), i.e., $V^n = \{V_i \mid 1 \leq i \leq n\}$. We also denote the domain of each grid as $D(u) = \{V_1, \dots, V_n\}$.

Thus, the aim of the LSC problem can be seen as finding a legal n -coloring where the number of vertices for each color class is exactly n . During the search process, V is used to denote the current set of color classes.

Reduction or Smart Solving

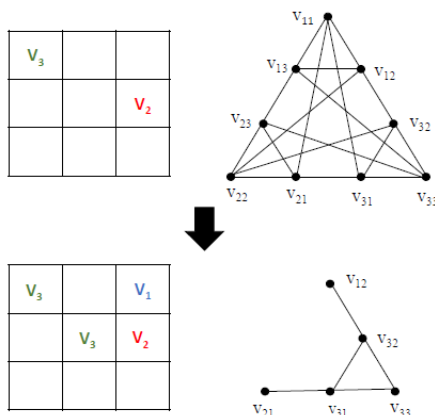
The paper proposes a reduction method which allows us to reduce the number of candidate variables. The following reduction rules are proposed in the paper:

Reduction Rule 1: If a vertex v has only one optional color class (i.e., $D(v) = \{V_i\}$), then vertex v should be put into the color class V_i , $D(v) = \emptyset$ and $D(u) = D(u) \setminus \{V_i\}$ for $\forall u \in N(v)$.

Reduction Rule 2: For a vertex v , $S_v^r = D(v) \setminus \bigcup_{u \in RN(v)} D(u)$. If the size of S_v^r is exactly one (i.e., $S_v^r = \{V_i\}$), then vertex v should be put into the only one color class V_i , $D(v) = \emptyset$ and $D(u) = D(u) \setminus \{V_i\}$ for $\forall u \in N(v)$.

Reduction Rule 3: For a vertex v , $S_v^c = D(v) \setminus \bigcup_{u \in CN(v)} D(u)$. If the size of S_v^c is exactly one (i.e., $S_v^c = \{V_i\}$), then vertex v should be put into the only one color class V_i , $D(v) = \emptyset$ and $D(u) = D(u) \setminus \{V_i\}$ for $\forall u \in N(v)$.

This reduction is analogous to how we solve sudoku. We don't start solving sudoku by trial and error, but by looking at those cells where answers can be obtained directly. For eg. if a row in latin square of order 3 already has 1 and 2, then the next cell has to be 3. This direct analysis allows us to reduce the candidate set and hence significantly reduce the time complexity and steps required.



The image depicts how reduction can allow us to reduce our graph coloring problem.

Consider cell (0,2). Here, we are sure that V_2 and V_3 can not be the answers. Hence, we immediately assign it V_1 .

The entire reduction runs in polynomial time or $O(n^3)$ time complexity.

CONFLICT VALUE SELECTION HEURISTIC

Consider the current color class set as V^n , and $V_k \in V^n$. If $v, u \in V_k$ and $(v, u) \in E$, then (v, u) is called a conflict edge. We use $CL(V_n)$ to denote the total number of conflict edges in V_n .

Moving a vertex v from color class V_i to another class V_j is denoted as $\text{Move}(v, V_i, V_j)$, which will result into a new color set class V_c^n

Hence, the heuristic is defined as follows

$$pscore(v, V_i, V_j) = CL(V^n) - CL(V_c^n)$$

Note that this heuristic intuitively reflects the effects of the moving operation for the current color class set

Hence, the selection rule is as follows: pick an operation $\text{Move}(v, V_i, V_j)$ with the biggest conflict heuristic value

GRAPH CONSTRUCTION

Algorithm 1: *Construct*(G)

Input: A Latin square graph $G = (V, E)$ where $|V| = n^2$

Output: The set of color classes V^n

```
1 initialize a color domain  $D(v)$  for  $\forall v \in V$ ;  
2 initialize a color class  $V_i$  for  $\forall V_i \in V^n$ ;  
3  $CandSet := V$ ;  
4 while  $CandSet$  is not empty do  
5   if there exists vertex  $v_1 \in CandSet$  satisfying any  
   reduction rule or as a filed grid then  
6      $V := V \setminus \{v_1\}$ ;  
7     put vertex  $v_1$  into a color class  $V_i$  based on  
     reduction rule;  
8      $CandSet := CandSet \setminus \{v_1\}$ ;  
9      $D(u) := D(u) \setminus \{V_i\}$  for  $\forall u \in N(v_1)$ ;  
10  else break ;  
11 while  $CandSet$  is not empty do  
12   select a random vertex  $v_2$  from  $CandSet$ ;  
13    $CandSet := CandSet \setminus \{v_2\}$ ;  
14   put vertex  $v_2$  into a random color class selected from  
    $D(v_2)$ ;  
15 return  $V^n$ ;
```

RESULTS

With State Space reduction

BFS

```
● verve@verve:~/Latin-Square-Completion$ g++ main.cpp
● verve@verve:~/Latin-Square-Completion$ ./a.out
Soln
Steps taken: 81
2 1 3
3 2 1
1 3 2
finished computation at Thu Dec 28 14:18:15 2023
elapsed time: 0.0296923s
```

DFS

```
● verve@verve:~/Latin-Square-Completion$ g++ main.cpp
● verve@verve:~/Latin-Square-Completion$ ./a.out
Soln
Steps taken: 50
2 3 1
3 1 2
1 2 3
finished computation at Thu Dec 28 14:18:25 2023
elapsed time: 0.0201633s
```

BestFS

```
● verve@verve:~/Latin-Square-Completion$ g++ main.cpp
● verve@verve:~/Latin-Square-Completion$ ./a.out
Soln
Steps taken: 3
2 1 3
3 2 1
1 3 2
finished computation at Thu Dec 28 14:19:41 2023
elapsed time: 0.00326363s
```

Without State Space reduction

BFS

```
● verve@verve:~/Latin-Square-Completion$ g++ main.cpp
● verve@verve:~/Latin-Square-Completion$ ./a.out
Soln
Steps taken: 59
2 3 1
3 1 2
1 2 3
finished computation at Thu Dec 28 14:29:58 2023
elapsed time: 0.0275391s
```

DFS

```
● verve@verve:~/Latin-Square-Completion$ g++ main.cpp
● verve@verve:~/Latin-Square-Completion$ ./a.out
Soln
Steps taken: 6917
1 3 2
2 1 3
3 2 1
finished computation at Thu Dec 28 14:29:33 2023
elapsed time: 2.06681s
```

BestFS

```
● verve@verve:~/Latin-Square-Completion$ g++ main.cpp
● verve@verve:~/Latin-Square-Completion$ ./a.out
Soln
Steps taken: 7
2 3 1
3 1 2
1 2 3
finished computation at Thu Dec 28 14:31:05 2023
elapsed time: 0.00582707s
```