Polynomial Time Construction for Spatially Balanced Latin Squares

Ronan Le Bras Andrew Perrault Carla P. Gomes

Department of Computer Science
Cornell University
Ithaca, NY 14853
{lebras,gomes}@cs.cornell.edu, arp86@cornell.edu

April 16, 2012

Abstract

In this paper we propose a polynomial time construction that generates *spatially balanced Latin squares* (SBLSs). These structures are central to the design of agronomic experiments, as they avoid biases that are otherwise unintentionally introduced due to spatial auto-correlation. Previous approaches were able to generate SBLSs of order up to 35 and required about two weeks of computation. Our algorithm runs in $\mathcal{O}(n^2)$ and generates SBLSs of arbitrary order n, where $2n+1$ is prime. For example, this algorithm generates a SBLS of order 999 in a fraction of a second.

1 Introduction

The theory and construction of combinatorial designs has raised several interesting research questions, Euler's conjecture [1, 2, e.g.] being one out of many. In this paper, we propose a polynomial time construction for *spatially balanced Latin squares*, confirming a conjecture formulated in 2004. Before defining the problem, we introduce the background motivation that led to its formulation.

In order to compare various soil treatments, agronomic laboratories must devise experimental designs that avoid the introduction of biases in the assessment of these treatments. For example, soil treatments encompass the use of fertilizers as well as different ways of preparing the soil. Fertilizers have both a high socio-economic and high environmental impact. Although they contribute to a low-cost high yield food production, fertilizers are responsible for massive water pollution, such as the one observed in the coastal waters of the Gulf of Mexico. Better experiment designs directly translate into a better assessment of the quality of fertilizers and their impact on the environment.

Most agronomic field experiments are typically designed in a randomized fashion, in what is called a randomized complete block design (RCBD) [3]. The RCBD methodology advocates dividing the experiments in blocks, where each block represents a replication and has as many experimental units as treatments, and assigning treatments at random within one block. However, even these randomized designs exhibit spatial dependence that is due to the autocorrelation of nearby treatments. Among others, the major nuisance factors that affect the assessment of the variables of interest (i.e. the efficiency of a fertilizer) are fertility patterns in fields, erosion as well as drainage variability. To address this issue, van Es and van Es [3] studied spatial distances in RCBDs and evaluated the impact of spatially unbalanced designs on the outcome of field experiments. They suggested to overcome the limitations of RCBDs through the use of *spatially balanced* designs. When the number of treatments equals the number of replications, the proposed structure is called a *spatially balanced Latin square* (SBLS).

In addition to field experiments, SBLSs can also be applied to greenhouse trials, chemical analyses involving multi-well titer plates and genomics research involving microarray slides [4].

The rest of the paper is organized as follows. We formally introduce spatially balanced Latin squares in the next section. In section 3, we presents related work. Section 4 describes our construction algorithm for SBLS (called `Unroll&Bounce`) while Section 5 reports the experimental results. Final comments are given in Section 6.

2 Preliminaries

A *Latin* square of order n on symbols 1 to n is an $n \times n$ square in which any symbol from 1 to n appears exactly once in each row and column. In order to be *spatially balanced*, a *Latin* square must also satisfy the following condition: for every pair of symbols, the number of columns separating the two symbols in a row summed over all rows is equal. Formally, we define $c(i, v)$ the column index where symbol v occurs in row i. The *distance* of the pair (v, v') of symbols in row i corresponds to $d(i, v, v') = |c(i, v) - c(i, v')|$. Finally, we define the *total distance* of a pair (v, v') as $d(v, v') = \sum_{i=1}^{n} d(i, v, v')$.

Definition 1. *A SBLS of order n is a Latin square of order n where the total distance is the same for any pair of symbols.*

Figure 1 depicts a SBLS of order 6 (which is actually the SBLS our algorithm generates). In this example, the total distance between symbols 1 and 2 is $d(1, 2) = |1 - 2| + |6 - 1| + |4 - 5| + |3 - 6| + |5 - 3| + |2 - 4| = 14$, the same for all pairs of symbols. The definition of a SBLS entails the following lemma.

Lemma 1. *There exists no SBLS of order $n = 3k + 1$, where $k \in \mathbb{N}^*$.*

Proof. In any row, there are exactly $n - d$ pairs of symbols that are d columns apart. Hence, the total distance available for all pairs of symbols in the square is $n \sum_{d=1}^{n-1} d(n - d) = \frac{n^2(n-1)(n+1)}{6}$. This needs to divide evenly into the $\binom{n}{2}$ pairs of symbols, so $\frac{n(n+1)}{3}$ must be an integer. Therefore, n cannot be congruent to 1 modulo 3. $\square$

As a direct consequence of this proof, the definition of a SBLS requires that $d(v, v') = \frac{n(n+1)}{3}$ for any pair (v, v') of symbols. For an instance of order 6 such as in Figure 1, the total distance of any two symbols equals $\frac{6(6+1)}{3} = 14$, which validates the value of $d(1, 2)$.

3 Related work

This work is concerned with the proof of existence and the construction of a special case of *Latin* squares. *Latin* square constructions have been a prolific topic in the literature of combinatorial design. For example, Heinrich et al.[5] provide a general construction for building *perfect Latin squares* of order n^2. In [6], Colbourn et al. give a detailed list of constructions for *mutually orthogonal Latin squares*. Finally, subsequent work such as [7] provides a construction for the only remaining possible configuration of *r-orthogonal Latin square*.

As with any new combinatorial design, it is legitimate to wonder how hard it is to complete a partial SBLS. The partial Latin square completion problem has been proven to be $\mathcal{NP}$-complete in [8] for symmetric squares, in [9] for arbitrary squares, and in [10] for symmetric squares where the partial quasigroup is constructed from a partial Steiner triple system. Whether the completion of a partial SBLS is $\mathcal{NP}$-complete remains an open question.

As discussed in the previous section, spatially balanced Latin squares (SBLSs) were first suggested in [3]. Gomes et al.[11] proposed both a simulated annealing (SA) approach and a constraint programming (CP) approach to solve this problem. The latter was able to generate SBLSs of order up to 9, while the former could solve SBLSs of order up to 12.

Introducing a new framework for boosting constraint reasoning, Gomes and Sellmann [12] solved SBLSs of order up to 18 by partitioning the solution space into subspaces with additional structure. This framework is called *streamlined constraint reasoning*. It is based on the observation that systematically maintaining all solutions will clearly limit the effectiveness of constraint propagation. Conversely, a subspace of solutions might be strongly characterized by an underlying structure. Hence, isolating such a subspace might dramatically boost constraint reasoning.

Smith et al.[13] extended the concept of streamlining to local search. Their approach confines a local search in the space of valid Latin squares, and defines a column swapping as a move in this space. This method is the current state of the art for solving the SBLS problem as it is capable of generating SBLSs of order up to 35 (with about two weeks of computation).

The following section presents our polynomial time construction for SBLS (`Unroll&Bounce`), which generates, for example, a SBLS of order 999 in 0.02 seconds.

4 Construction

In this section, we present the algorithm `Unroll&Bounce` which generates a SBLS of any order n where $2n+1$ is prime. We then derive a formula for filling any cell of the SBLS resulting from the algorithm. Finally, we prove the correctness of this construction.

4.1 Algorithm `Unroll&Bounce`

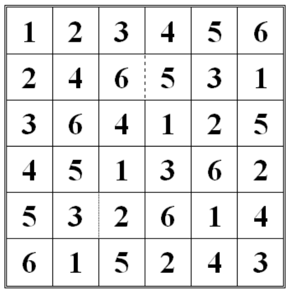
Our algorithm proceeds row by row. The intuition is to fill every row i ($i = 1, ..., n$) of the square, starting with i and successively adding i to the previous cell. In a sense, we unroll the set of symbols that are 0 (mod i), which we call the first *group*. Once we reach the largest multiple v of i that is lower or equal to n, we perform what we call an *upper bounce*: the next cell is assigned symbol $2n+1-i-v$. This symbol is the starting point for the second group. We unroll this group by successively subtracting i until we reach a value v' that is lower or equal to i. At this point, we perform a *lower bounce*: the next cell is assigned symbol $i-v'$. This symbol represents the starting point of the next group and we repeat these steps until the row is full.

In the following, we denote A_n the square of order n generated by our algorithm, and equivalently a_{ij} or $a[i,j]$ the symbol in row i and column j of the square A_n.

```
for rows $i = 1, 2, \ldots, n$ do
    $k = 1$; $j = 1$;                                   // Group counter and column index
    $a[i.j] = i$;                                        // First symbol of row i
    while $j < n$ do
        /* Unroll symbols in group k */
        if $k$ is odd then
            while $a[i,j] + i \leq n$ and $j < n$ do
                $a[i,j+1] = a[i,j] + i$;                 // Odd group
                $j = j + 1$;
        else
            while $a[i,j] - i \geq 1$ and $j < n$ do
                $a[i,j+1] = a[i,j] - i$;                 // Even group
                $j = j + 1$;
        /* Perform a bounce */
        if $j < n$ then
            if $k$ is odd then
                $a[i,j+1] = 2n + 1 - i - a[i,j]$;        // Upper bounce
            else
                $a[i,j+1] = i - a[i,j]$;                 // Lower bounce
            $k = k + 1$;
            $j = j + 1$;
```

Algorithm 1: Construction `Unroll&Bounce` for SBLS of order n.

This algorithm is described in Algorithm 1 and illustrated in Figure 1. In row 2, we start with value 2 and we unroll the first group ($\{2, 4, 6\}$), i.e. all symbols that are 0 (mod 2). We then perform an upper bounce (as the dashed line illustrates). The next cell therefore is assigned value $2n + 1 - i - 6 = 5$. Afterwards, we unroll the second group ($\{5, 3, 1\}$), i.e. the symbols that are congruent to $2n+1$ (mod 2). Similarly, in row 5, the dotted line illustrates a lower bounce between columns 2 and 3. After value 3 comes value 2, as $i - 3 = 2$. The corresponding groups of each row are depicted in Figure 2. For instance, in row 2, the first group ($\{2, 4, 6\}$) covers the first three cells, and the second ($\{5, 3, 1\}$) covers the other three.



1	2	3	4	5	6
2	4	6	5	3	1
3	6	4	1	2	5
4	5	1	3	6	2
5	3	2	6	1	4
6	1	5	2	4	3

Figure 1: A_6, SBLS of order $n = 6$ generated by the `Unroll&Bounce` algorithm.

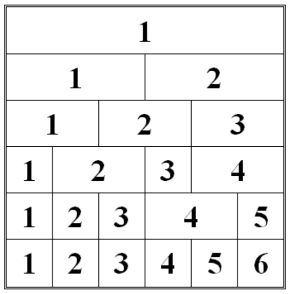


Figure 2: Groups k_{ij} for A_6: two adjacent cells are merged if they belong to the same group.

The complexity of Algorithm 1 is $\mathcal{O}(n^2)$. Note that checking the validity of the resulting matrix requires cubic time in the size of the square.

4.2 Formula

More formally, we can derive a formula that summarizes the previous algorithm. Let us define k_{ij} as the group the cell (i, j) belongs to, i.e. one more than the number of bounces that occur before column j in row i. We can prove, by induction on the group, the following equation:

$$a_{ij} = (-1)^{k_{ij}+1} \left(ij - \lfloor \frac{k_{ij}}{2} \rfloor (2n+1) \right) \tag{1}$$

The derivation of an analytic formula for k_{ij} is less straightforward. We first derive a formula for the last term of any given group in any row. Suppose k_{ij} is odd. By construction, the last value of this group has to verify:

$$\begin{aligned} n - i <& ij - \lfloor \frac{k_{ij}}{2} \rfloor (2n+1) \leq n \\ \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1) + n}{i} - 1 <& j \leq \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1) + n}{i} \\ j =& \lfloor \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1) + n}{i} \rfloor \end{aligned} \tag{2}$$

Similarly, if k_{ij} is even and a_{ij} corresponds to the last value of this group, it yields:

$$\begin{aligned} 1 \leq& \lfloor \frac{k_{ij}}{2} \rfloor (2n+1) - ij \leq i \\ \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1)}{i} - 1 \leq& j \leq \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1) - 1}{i} \\ \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1)}{i} - 1 \leq& j \leq \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1)}{i} \\ \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1)}{i} - 1 <& j \leq \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1)}{i} && (3) \\ j =& \lfloor \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1)}{i} \rfloor && (4) \end{aligned}$$

Note that the strict inequality in (3) holds since *i*) $\lfloor \frac{k_{ij}}{2} \rfloor < i$ (as we would otherwise have $(2n+1) - 1 \leq j$) and *ii*) $2n+1$ is prime. These two conditions ensure that $\frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1)}{i}$ cannot be an integer value, as opposed to j.

Equations 2 and 4 provide the column indices after which an upper bounce and a lower bounce occur, respectively. In other words, the first group of row i covers column 1 to column $\lfloor \frac{n}{i} \rfloor$, the second group spreads to column $\lfloor \frac{2n+1}{i} \rfloor$, the third group to column $\lfloor \frac{3n+1}{i} \rfloor$, etc. For example, as illustrated in Figure 2, in row 4 of our running example, the first group goes up to column $\lfloor \frac{6}{4} \rfloor = 1$, the second up to column $\lfloor \frac{2*6+1}{4} \rfloor = 3$, the third up to column $\lfloor \frac{3*6+1}{4} \rfloor = 4$, and the fourth up to column $\lfloor \frac{4*6+2}{4} \rfloor = 6$.

Now, let us take an arbitrary cell (i, j) of A_n that belongs to an odd group k_{ij}. The column j has to be greater than the last column of group $k_{ij} - 1$, and less or equal to the last column of group k_{ij}. Using equations 2 and 4 and setting $k_{ij} = 2l_{ij} + 1$, we obtain:

$$\begin{aligned}
\lfloor \frac{\lfloor \frac{k_{ij}-1}{2} \rfloor (2n+1)}{i} \rfloor &< j \leq \lfloor \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1) + n}{i} \rfloor \\
\lfloor \frac{l_{ij}(2n+1)}{i} \rfloor &< j \leq \lfloor \frac{l_{ij}(2n+1) + n}{i} \rfloor \\
\frac{l_{ij}(2n+1)}{i} &< j \leq \frac{l_{ij}(2n+1) + n}{i}
\end{aligned}$$

This last equation is derived from the previous one using the fact that j is an integer as opposed to $\frac{l_{ij}(2n+1)}{i}$. At this point, we want to express l_{ij} as a function of i and j, which yields:

$$\begin{aligned}
\frac{ij-n}{2n+1} &\leq l_{ij} < \frac{ij}{2n+1} \\
\frac{2ij}{2n+1} - \frac{2n}{2n+1} &\leq 2l_{ij} < \frac{2ij}{2n+1} \\
\frac{2ij}{2n+1} - 1 &< 2l_{ij} < \frac{2ij}{2n+1} \\
2l_{ij} &= \lfloor \frac{2ij}{2n+1} \rfloor \\
k_{ij} &= \lfloor \frac{2ij}{2n+1} \rfloor + 1, \text{ where } k_{ij} \text{ is odd.} \qquad (5)
\end{aligned}$$

Similarly, we take an arbitrary cell (i, j) that belongs to an even group k_{ij}. We set $k_{ij} = 2l_{ij}$ and we obtain:

$$\begin{aligned}
\lfloor \frac{\lfloor \frac{k_{ij}-1}{2} \rfloor (2n+1) + n}{i} \rfloor &< j \leq \lfloor \frac{\lfloor \frac{k_{ij}}{2} \rfloor (2n+1)}{i} \rfloor \\
\frac{(l_{ij}-1)(2n+1) + n}{i} &< j \leq \frac{l_{ij}(2n+1)}{i}
\end{aligned}$$

Again, expressing l_{ij} as a function of i and j yields:

$$\begin{aligned}
\frac{ij}{2n+1} &\leq l_{ij} < \frac{ij-n}{2n+1} + 1 \\
\frac{2ij}{2n+1} &\leq 2l_{ij} < \frac{2ij-2n}{2n+1} + 2 \\
\frac{2ij}{2n+1} &< 2l_{ij} < \frac{2ij+1}{2n+1} + 1
\end{aligned}$$

The strict inequality at line 3 comes from the fact that $2n+1$ is prime. Now, we would like to make the right hand side tighter, so that we end up again with Equation 5, but for k_{ij} even. If $\frac{2ij+1}{2n+1} = \lfloor \frac{2ij+1}{2n+1} \rfloor$, then the right hand side yields $2l_{ij} < \lfloor \frac{2ij+1}{2n+1} \rfloor + 1$. This leads to $2l_{ij} \leq \lfloor \frac{2ij+1}{2n+1} \rfloor = \frac{2ij+1}{2n+1} \leq \frac{2ij}{2n+1} + 1$. Conversely, if $\frac{2ij+1}{2n+1} > \lfloor \frac{2ij+1}{2n+1} \rfloor$, then $\frac{2ij+1}{2n+1} \geq \lfloor \frac{2ij+1}{2n+1} \rfloor + \frac{1}{2n+1}$ and the right hand side yields $2l_{ij} \leq \lfloor \frac{2ij+1}{2n+1} \rfloor + 1 \leq \frac{2ij+1}{2n+1} - \frac{1}{2n+1} + 1 \leq \frac{2ij}{2n+1} + 1$. Thus, both cases lead to:

$$\begin{aligned}
\frac{2ij}{2n+1} &< 2l_{ij} \leq \frac{2ij}{2n+1} + 1 \\
k_{ij} &= \lfloor \frac{2ij}{2n+1} \rfloor + 1, \text{ where } k_{ij} \text{ is even.} \qquad (6)
\end{aligned}$$

Hence, by combining Equations 5 and 6, we actually derive Equation 7:

$$k_{ij} = \lfloor \frac{2ij}{2n+1} \rfloor + 1 \tag{7}$$

Using equations 1 and 7, we can directly derive any value of the square generated by our algorithm `Unroll&Bounce`. For instance, we can retrieve the value of cell $(2,4)$ in the SBLS illustrated in Figure 1 as follows. First, this cell belongs to the group $k_{2,4} = \lfloor \frac{2*2*4}{2*6+1} \rfloor + 1 = 2$. Next, we use Equation 1 to compute its value: $a[2,4] = (-1)^{2+1} \left(2*4 - \lfloor \frac{2}{2} \rfloor (2*6+1)\right) = (-1)(8-13) = 5$. Now we prove that our construction does produce a spatially balanced *Latin* square (SBLS).

4.3 Latin property

We first prove that the construction generates a square that ensures the *Latin* property (Theorem 1), namely that each value from 1 to n appears exactly once in each row and column.

Lemma 2. *For any order n where $2n+1$ is prime, every symbol from 1 to n appears exactly once in each row of A_n.*

Proof. First, let us take two symbols a_{ij_1} and a_{ij_2} that belong to two different groups in row i, i.e., $k_{ij_1} \neq k_{ij_2}$. Arbitrarily, we set $k_{ij_1} < k_{ij_2}$. Suppose, by way of contradiction, that $a_{ij_1} = a_{ij_2}$. Then, $a_{ij_1} \equiv a_{ij_2} \pmod{i}$, which leads to $(-1)^{k_{ij_1}} \lfloor \frac{k_{ij_1}}{2} \rfloor (2n+1) \equiv (-1)^{k_{ij_2}} \lfloor \frac{k_{ij_2}}{2} \rfloor (2n+1) \pmod{i}$, by Equation 1. Therefore, we have $(2n+1)\left((-1)^{k_{ij_1}} \lfloor \frac{k_{ij_1}}{2} \rfloor - (-1)^{k_{ij_2}} \lfloor \frac{k_{ij_2}}{2} \rfloor\right) \equiv 0 \pmod{i}$. As $2n+1$ is prime, i divides $(-1)^{k_{ij_1}} \lfloor \frac{k_{ij_1}}{2} \rfloor - (-1)^{k_{ij_2}} \lfloor \frac{k_{ij_2}}{2} \rfloor$. Moreover, $1 \leq k_{ij_1} < k_{ij_2} \leq i$ implies that $-i < (-1)^{k_{ij_1}} \lfloor \frac{k_{ij_1}}{2} \rfloor - (-1)^{k_{ij_2}} \lfloor \frac{k_{ij_2}}{2} \rfloor < i$. As a result, we have $(-1)^{k_{ij_1}} \lfloor \frac{k_{ij_1}}{2} \rfloor = (-1)^{k_{ij_2}} \lfloor \frac{k_{ij_2}}{2} \rfloor$. This leads us to a contradiction, since $k_{ij_1} \neq k_{ij_2}$.

In addition, we know by construction that two symbols within a group are distinct. Thus, all symbols in a row are different. The fact that, by construction, any symbol is between 1 and n and any row contains n symbols completes the proof. □

Theorem 1 (Latin property)**.** *For any order n where $2n+1$ is prime, A_n is a Latin square.*

Proof. It follows from Equation 7 that $k_{ij} = k_{ji}$. Hence, we deduce from Equation 1 that $a_{ij} = a_{ji}$. In other words, `Unroll&Bounce` produces a symmetric square. As a result, Lemma 2 implies that every number from 1 to n appears exactly once in each column as well. □

4.4 Row composition commutativity

In this section we prove an important property of the square generated with our construction. In [12], Gomes et al. noticed from the samples they were able to generate that many SBLSs share an important underlying property, namely they are *self-symmetric*. As defined in [12], the term self-symmetric denotes a square that is symmetric to itself when applying any combination of row and symbol permutations that preserves the first row and column. In the special case where the square is a symmetric SBLS, as the one generated with our algorithm, the self-symmetry is equivalent to row composition commutativity. We define row composition commutativity as follows:

Definition 2. *For any square $(a[x,y])_{1\leq x,y\leq n}$, the row composition is commutative if and only if $a[x,a[y,z]] = a[y,a[x,z]],\ \forall 1 \leq x,y,z \leq n$.*

To illustrate this property with the example in Figure 1, consider for instance rows 2 and 4. If we focus on column 5, we have $a[2,5] = 3$ and $a[4,5] = 6$. Literally, row composition commutativity means that $a[4,3]$ and $a[2,6]$ must have the same value. Indeed, they are both equal to 1 in our example. In order to consider all columns at once and for ease of notation, we denote by σ_i the permutation induced by row i. We have $\sigma_2 = (2,4,6,5,3,1)$ and $\sigma_4 = (4,5,1,3,6,2)$. By composition of these two rows, we obtain $\sigma_4 \circ \sigma_2 = \sigma_2 \circ \sigma_4 = (5,3,2,6,1,4)$, which demonstrates that the two rows do commutate. In the following, we formally prove the row composition commutativity of the square generated with our algorithm.

Lemma 3. *For any order n where $2n+1$ is prime, A_n is a row-composition commutative square.*

Proof. From Equations 1 and 7, and adopting the notation in Definition 2, every entry $a[x,y]$ can be written either as $a[x,y] = xy - l_{xy}(2n+1)$ (if k_{xy} is odd with $k_{xy} = 2l_{xy}+1$) or as $a[x,y] = l_{xy}(2n+1) - xy$ (if k_{xy} is even with $k_{xy} = 2l_{xy}$). If k_{xy} is odd, $k_{a[x,y],z}$ can be written as follows:

$$\begin{aligned} k_{xy-l_{xy}(2n+1),z} &= \lfloor \frac{2xyz - 2l_{xy}(2n+1)z}{2n+1} \rfloor + 1 \\ &= \lfloor \frac{2xyz}{2n+1} \rfloor + 1 - 2l_{xy}z \\ &= k_{xy,z} - 2l_{xy}z \end{aligned}$$

Otherwise, if k_{xy} is even, since $2n+1$ prime implies $\lfloor \frac{-2xyz}{2n+1} \rfloor = -\lfloor \frac{2xyz}{2n+1} \rfloor - 1$ (as $1 \leq x,y,z \leq n$), we obtain:

$$\begin{aligned} k_{l_{xy}(2n+1)-xy,z} &= \lfloor \frac{2l_{xy}(2n+1)z - 2xyz}{2n+1} \rfloor + 1 \\ &= 2l_{xy}z - \lfloor \frac{2xyz}{2n+1} \rfloor - 1 + 1 \\ &= 2l_{xy}z - k_{xy,z} + 1 \end{aligned}$$

For the sake of completeness, we now consider three different cases, which are combinations of possibilities for k_{xz} and k_{yz}. In the second and third cases, we use the fact that $\lfloor \frac{-x+1}{2} \rfloor = -\lfloor \frac{x}{2} \rfloor$ for any integer x.

- Case 1. $a[x,z] = xz - l_{xz}(2n+1)$ and $a[y,z] = yz - l_{yz}(2n+1)$:

$$\begin{aligned} a[x,a[y,z]] &= (-1)^{k_{yz,x}-2l_{yz}x+1}[xyz - l_{yz}x(2n+1) - \lfloor \frac{k_{yz,x} - 2l_{yz}x}{2} \rfloor (2n+1)] \\ &= (-1)^{k_{yz,x}+1}[xyz - \lfloor \frac{k_{yz,x}}{2} \rfloor (2n+1)] \\ a[y,a[x,z]] &= (-1)^{k_{xz,y}-2l_{xz}y+1}[xyz - l_{xz}y(2n+1) - \lfloor \frac{k_{xz,y} - 2l_{xz}y}{2} \rfloor (2n+1)] \\ &= (-1)^{k_{xz,y}+1}[xyz - \lfloor \frac{k_{xz,y}}{2} \rfloor (2n+1)] \end{aligned}$$

- Case 2. $a[x,z] = xz - l_{xz}(2n+1)$ and $a[y,z] = l_{yz}(2n+1) - yz$:

$$\begin{aligned}
a[x,a[y,z]] \quad &= (-1)^{2l_{yz}x - k_{yz,x}+2}[l_{yz}x(2n+1) - xyz - \lfloor\frac{2l_{yz}x - k_{yz,x}+1}{2}\rfloor(2n+1)] \\
&= (-1)^{k_{yz,x}}[-xyz + \lfloor\frac{k_{yz,x}}{2}\rfloor(2n+1)] \\
a[y,a[x,z]] \quad &= (-1)^{k_{xz,y} - 2l_{xz}y+1}[xyz - l_{xz}y(2n+1) - \lfloor\frac{k_{xz,y} - 2l_{xz}y}{2}\rfloor(2n+1)] \\
&= (-1)^{k_{xz,y}+1}[xyz - \lfloor\frac{k_{xz,y}}{2}\rfloor(2n+1)]
\end{aligned}$$

- Case 3. $a[x,z] = l_{xz}(2n+1) - xz$ and $a[y,z] = l_{yz}(2n+1) - yz$:

$$\begin{aligned}
a[x,a[y,z]] \quad &= (-1)^{2l_{yz}x - k_{yz,x}+2}[l_{yz}x(2n+1) - xyz - \lfloor\frac{2l_{yz}x - k_{yz,x}+1}{2}\rfloor(2n+1)] \\
&= (-1)^{k_{yz,x}}[-xyz + \lfloor\frac{k_{yz,x}}{2}\rfloor(2n+1)] \\
a[y,a[x,z]] \quad &= (-1)^{2l_{xz} - k_{xz,y}+2}[l_{xz}y(2n+1) - xyz - \lfloor\frac{2l_{xz}y - k_{xz,y}+1}{2}\rfloor(2n+1)] \\
&= (-1)^{k_{xz,y}}[-xyz + \lfloor\frac{k_{xz,y}}{2}\rfloor(2n+1)]
\end{aligned}$$

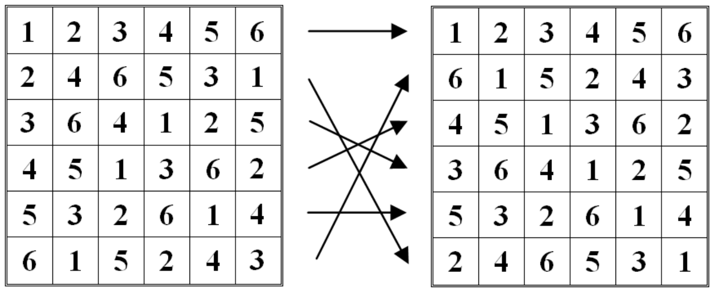
Therefore, in any of these three cases, as $k_{xz,y} = k_{yz,x}$, we have $a[x,a[y,z]] = a[y,a[x,z]]$, which proves the row composition commutativity of the square generated by `Unroll&Bounce`. $\square$

Whereas most SBLSs share the self-symmetric property, this property is not a sufficient condition. Consequently, the balancedness property remains to be proved.

4.5 Balancedness property

Proving balancedness happens to be even more challenging than proving the *Latin* property or the row composition commutativity. We successively demonstrate interesting additional properties, that we call *class property*, *dist-min relation* and *min rule*, which constitute steps towards proving balancedness of A_n.

Along this proof, we reason on the *column conjugate* $\tilde{A}_n$ of A_n. By definition, any value a_{ij} of A_n corresponds to the column in which value j appears in row i in $\tilde{A}_n$. In the following, we add a tilde on a term when it refers to $\tilde{A}_n$ instead of A_n. Figure 3 depicts $\tilde{A}_6$.



1	2	3	4	5	6
2	4	6	5	3	1
3	6	4	1	2	5
4	5	1	3	6	2
5	3	2	6	1	4
6	1	5	2	4	3

1	2	3	4	5	6
6	1	5	2	4	3
4	5	1	3	6	2
3	6	4	1	2	5
5	3	2	6	1	4
2	4	6	5	3	1

Figure 3: A_6 (left) and its column conjugate $\tilde{A}_6$ (right) are in fact identical up to a row permutation.

We observe that there exists a permutation of rows that transforms $\tilde{A}_n$ into A_n. Lemma 4 formalizes and demonstrates the aforementioned observation.

Lemma 4. $\forall\ 1 \leq q \leq n,\ \tilde{\sigma}_i = \sigma_{\tilde{a}[i,1]}$

Proof. First, it follows from the definition of the column conjugate that $\tilde{\sigma}_i = \sigma_i^{-1}$, since $\sigma_i(j) = v$ if and only if $\tilde{\sigma}_i(v) = j$. Second, we note that A_n is in *reduced* form: both the first row and the first column correspond to $(1, 2, ..., n)$. For any row q and any symbol v, we have:

$$\begin{aligned}
\tilde{\sigma}_i^{-1} \circ \sigma_{\tilde{a}[i,1]}(v) &= \sigma_i \circ \sigma_{\tilde{a}[i,1]}(v) && \text{(column conjugate property)}\\
&= \sigma_i \circ \sigma_v(\tilde{a}[i,1]) && \text{(diagonal symmetry)}\\
&= \sigma_v \circ \sigma_i(\tilde{a}[i,1]) && \text{(row comp. commutativity)}\\
&= \sigma_v \circ \sigma_i(\tilde{\sigma}_i(1)) &&\\
&= \sigma_v \circ \sigma_i(\sigma_i^{-1}(1)) && \text{(column conjugate property)}\\
&= \sigma_v(1) &&\\
&= v && \text{(reduced form of } A_n\text{)}
\end{aligned}$$

Hence, $\tilde{\sigma}_i(v) = \sigma_{\tilde{a}[i,1]}(v)$ for any row i and any symbol v. □

Additionally, this proof actually demonstrates that symmetry and row composition commutativity are a sufficient condition for a reduced *Latin* square to be a permutation of rows of its column conjugate.

We now prove that $\tilde{A}_n$ is *balanced*, which will imply that A_n is *balanced*, as a row permutation does not affect the balancedness of a matrix. From the definition of the column conjugate, the distance between a pair of symbols (v, v') in row i of $\tilde{A}_n$ corresponds to the absolute difference between symbols in column v and in column v' in row i of A_n. Formally, $\tilde{d}(i, v, v') = |\tilde{c}(i, v) - \tilde{c}(i, v')| = |a[i, v] - a[i, v']|$.

We first prove that there exist *classes* of pairs of symbols such that their total distance is decomposed similarly for any pair within its class. Table 1 shows the distances between symbols in $\tilde{A}_6$. We observe that the pairs of symbols $\{(1, 2), (1, 6), (2, 4), (3, 5), (3, 6), (4, 5)\}$ belong to a class where the distances between symbols correspond to the set $\{1, 1, 2, 2, 3, 5\}$. Similarly, the pairs $\{(1, 3), (1, 4), (2, 5), (2, 6), (3, 4), (5, 6)\}$ belong to another class, and the pairs $\{(1, 5), (2, 3), (4, 6)\}$ belong to yet another class. This observation leads us to the following *class property*.

Table 1: Distances between pairs of symbols in $\tilde{A}_6$

j_1,j_2	1,2	1,3	1,4	1,5	1,6	2,3	2,4	2,5	2,6	3,4	3,5	3,6	4,5	4,6	5,6
$\tilde{d}(1, j_1, j_2)$	1	2	3	4	5	1	2	3	4	1	2	3	1	2	1
$\tilde{d}(2, j_1, j_2)$	2	4	3	1	1	2	1	1	3	1	3	5	2	4	2
$\tilde{d}(3, j_1, j_2)$	3	1	2	1	2	2	5	4	1	3	2	1	1	4	3
$\tilde{d}(4, j_1, j_2)$	1	3	1	2	2	4	2	1	3	2	5	1	3	1	4
$\tilde{d}(5, j_1, j_2)$	2	3	1	4	1	1	3	2	1	4	1	2	5	2	3
$\tilde{d}(6, j_1, j_2)$	5	1	4	2	3	4	1	3	2	3	1	2	2	1	1
$\tilde{d}(j_1, j_2)$	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14

Property 1 (Class property)**.** *For any pair of columns* (j_1, j_2)*,* $\tilde{d}(j_1, j_2) = \tilde{d}(\sigma_q(j_1), \sigma_q(j_2))$ $\forall q \in \{1, ..., n\}$

Proof. In this proof, we successively use the facts that (a) A_n is a *Latin* square (hence any row defines a permutation) (b) A_n is symmetric, and (c) A_n is row-composition commutative.

For any integer q in $\{1..n\}$, we have:

$$\begin{aligned}
\tilde{d}(j_1, j_2) &= \sum_i |a[i, j_1] - a[i, j_2]| \\
&= \sum_i |a[\sigma_q(i), j_1] - a[\sigma_q(i), j_2]| && \text{(a)} \\
&= \sum_i |a[a[q, i], j_1] - a[a[q, i], j_2]| \\
&= \sum_i |a[j_1, a[i, q]] - a[j_2, a[i, q]]| && \text{(b)} \\
&= \sum_i |a[i, a[j_1, q]] - a[i, a[j_2, q]]| && \text{(c)} \\
&= \sum_i |a[i, a[q, j_1]] - a[i, a[q, j_2]]| && \text{(b)} \\
&= \sum_i |a[i, \sigma_q(j_1)] - a[i, \sigma_q(j_2)]| \\
&= \tilde{d}(\sigma_q(j_1), \sigma_q(j_2))
\end{aligned}$$

□

Consequently, this property ensures that some pairs of columns share the same total symbol difference. In the following, we use this property to prove that the total distance between any two symbols is indeed equal to $\frac{n(n+1)}{3}$.

An important notion for the remaining of the proof is what we call *sum of minima of two columns.* Formally, we define it as:

$$\Sigma min(j_1, j_2) = \sum_{i=1}^{n} min(a[i, j_1], a[i, j_2])$$

For example, the sum of minima of columns 1 and 3 in A_6 is: $\Sigma min(1, 3) = 1+2+3+1+2+5 = 14$. Next, we derive a direct relationship between $\Sigma min(j_1, j_2)$ and $\tilde{d}(j_1, j_2)$.

Property 2 (Dist-min relation)**.** $\Sigma min(j_1, j_2) = \frac{n(n+1)}{2} - \frac{1}{2}\tilde{d}(j_1, j_2)$

Proof. Using the fact that $\tilde{d}(j_1, j_2) = 2 \sum_{i:a[i,j_1]<a[i,j_2]} \big(a[i, j_2] - a[i, j_1]\big)$ since $\sum_i a[i, j_2] - a[i, j_1] = 0$ for any *Latin* square, we have:

$$
\begin{aligned}
\Sigma min(j_1,j_2) &= \sum_{i:a[i,j_1]<a[i,j_2]} a[i,j_1] + \sum_{i:a[i,j_2]<a[i,j_1]} a[i,j_2] \\
&= \sum_{i:a[i,j_1]<a[i,j_2]} a[i,j_1] + \sum_{i} a[i,j_2] - \sum_{i:a[i,j_1]<a[i,j_2]} a[i,j_2] \\
&= \frac{n(n+1)}{2} - \sum_{i:a[i,j_1]<a[i,j_2]} a[i,j_1] - a[i,j_2] \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\tilde{d}(j_1,j_2)
\end{aligned}
$$

□

The use of this notion of *minimum of two columns* results from the following insight. For any pair p_1 of columns, there exists another pair p_2 such that the symbol difference of p_1 corresponds to the minimum of p_2 in any row. Table 2 illustrates this observation for the pairs $(1,2)$, $(2,5)$ and $(3,6)$ in our example of Figure 1, which correspond to the minimum of columns $(1,3)$, $(3,6)$ and $(3,4)$ respectively. We formalize this observation with the next property.

Table 2: Illustrating the *min rule* for three pairs of columns of A_6

i	$\tilde{d}(i,1,2)$	$min(a_{i,1},a_{i,3})$	$\tilde{d}(i,2,5)$	$min(a_{i,3},a_{i,6})$	$\tilde{d}(i,3,6)$	$min(a_{i,3},a_{i,4})$
1	1	1	3	3	3	3
2	2	2	1	1	5	5
3	3	3	4	4	1	1
4	1	1	1	1	1	1
5	2	2	2	2	2	2
6	5	5	3	3	2	2

Property 3 (Min rule)**.**

$$\forall 1 \le i, j_1, j_2 \le n, j_1 \ne j_2,\ \ \tilde{d}(i,j_1.j_2) = \begin{cases} min(a[i,|j_1-j_2|], a[i,j_1+j_2]) & \textit{if } j_1+j_2 \le n \\ min(a[i,|j_1-j_2|], a[i,2n+1-j_1-j_2]) & \textit{otherwise} \end{cases}$$

The proof of this property requires the analysis of 32 different cases, which correspond to combinations of truth values of the following propositions:
$\mathcal{A} : j_1+j_2 \le n$, $\mathcal{B} : a[i,j_1] < a[i,j_2]$, $\mathcal{C} : k_{ij_1}$ is odd, $\mathcal{D} : k_{ij_2}$ is odd, $\mathcal{E} : a[i,j_1]+a[i,j_2] \le n$. Without loss of generality, we assume $\mathcal{F} : j_1 > j_2$ holds.

We outline the proof below, providing two canonical cases. The proof makes extensive use of the following fact:

Fact 1. *For any $1 \le i,j \le n$ and $2n+1$ prime, there exists a unique integer l such that $1 \le ij - l(2n+1) \le n$ or $1 \le l(2n+1) - ij \le n$.*

- Case 1. $\mathcal{A},\mathcal{C},\mathcal{D},\mathcal{E}$ are true; $\mathcal{B}$ is false.

$$
\begin{aligned}
\tilde{d}(i,j_1,j_2) &= a[i,j_1] - a[i,j_2] && (\mathcal{B})\\
&= ij_1 - (2n+1)\lfloor\frac{k_{ij_1}}{2}\rfloor - ij_2 + (2n+1)\lfloor\frac{k_{ij_2}}{2}\rfloor && (\mathcal{C},\mathcal{D})\\
&= i(j_1-j_2) - (2n+1)(\lfloor\frac{k_{ij_1}}{2}\rfloor - \lfloor\frac{k_{ij_2}}{2}\rfloor) &&\\
&= a[i,j_1-j_2] && (\mathcal{F}, \text{Fact 1})\\
a[i,j_1]+a[i,j_2] &= i(j_1+j_2) - (2n+1)(\lfloor\frac{k_{ij_1}}{2}\rfloor + \lfloor\frac{k_{ij_2}}{2}\rfloor) && (\mathcal{C},\mathcal{D})\\
&= a[i,j_1+j_2] && (\mathcal{A},\mathcal{E}, \text{Fact 1})
\end{aligned}
$$

Therefore, the value in row i and column $j_1 + j_2$ corresponds to the sum of $a[i,j_1]$ and $a[i,j_2]$ while the value in column $j_1 - j_2$ corresponds to the difference. As a result, $a[i,j_1-j_2] \le a[i,j_1+j_2]$ and we conclude that $\tilde{d}(i,j_1,j_2) = a[i,j_1-j_2] = \Sigma min(a[i,j_1-j_2], a[i,j_1+j_2])$.

- Case 2. $\mathcal{A},\mathcal{C},\mathcal{E}$ are false; $\mathcal{B},\mathcal{D}$ are true.

$$
\begin{aligned}
\tilde{d}(i,j_1,j_2) &= a[i,j_2] - a[i,j_1] && (\mathcal{B})\\
&= ij_2 - (2n+1)\lfloor\frac{k_{ij_2}}{2}\rfloor - ((2n+1)\lfloor\frac{k_{ij_2}}{2}\rfloor - ij_2) && (\mathcal{C},\mathcal{D})\\
&= -(2n+1)i - i(j_1+j_2) + &&\\
&(2n+1)(\lfloor\frac{k_{ij_1}}{2}\rfloor - \lfloor\frac{k_{ij_2}}{2}\rfloor) + (2n+1)i &&\\
&= -i(2n+1-j_1-j_2) + (2n+1)(i - \lfloor\frac{k_{ij_1}}{2}\rfloor - \lfloor\frac{k_{ij_2}}{2}\rfloor) &&\\
&= a[i,2n+1-j_1-j_2] && (\mathcal{A}, \text{Fact 1})\\
2n+1-a[i,j_1]-a[i,j_2] &= 2n+1-(2n+1)\lfloor\frac{k_{ij_1}}{2}\rfloor + ij_1 - ij_2 + (2n+1)\lfloor\frac{k_{ij_2}}{2}\rfloor && (\mathcal{C},\mathcal{D})\\
&= 2n+1+i(j_1-j_2) - (2n+1)(\lfloor\frac{k_{ij_1}}{2}\rfloor - \lfloor\frac{k_{ij_2}}{2}\rfloor) &&\\
&= i(j_1-j_2) - (2n+1)(\lfloor\frac{k_{ij_1}}{2}\rfloor - \lfloor\frac{k_{ij_2}}{2}\rfloor - 1) &&\\
&= a[i,j_1-j_2] && (\mathcal{E},\mathcal{F}, \text{Fact 1})
\end{aligned}
$$

Again, column $2n+1-j_1-j_2$ contains the difference of $a[i,j_1]$ and $a[i,j_2]$ while column $j_1 - j_2$ contains $2n+1$ minus their sum. Moreover, the fact that $\mathcal{E}$ is false makes the latter value smaller than the former. Therefore, the minimum of the values of these two columns is the symbol difference between columns j_1 and j_2. The remaining 30 cases might be derived in a similar fashion.

Theorem 2 (Balancedness property)**.** *For any order n where $2n+1$ is prime, A_n is a balanced square.*

Proof. The proof goes as follows. Starting with an arbitrary pair p_1 of columns of A_n, the *min rule* establishes a direct relationship between this pair and a second pair p_2. Using the *dist-min* relation and applying the *min rule* again draws a relationship between p_2 and a third pair p_3. We then prove that p_1 and p_3 actually belong to the same class, which allows us to deduce the actual value of the symbol difference between the columns of p_1.

For example, let us consider the pair of columns $(2,5)$. As depicted in Table 2, $\tilde{d}(2,5) = \Sigma min(3,6)$. Moreover, Property 2 yields $\Sigma min(3,6) = 21 - \frac{1}{2}\tilde{d}(3,6)$. Applying the *min rule*, we obtain $\tilde{d}(3,6) = \Sigma min(3,4)$. In turn, Property 2 provides the equation $\Sigma min(3,4) = 21 - \frac{1}{2}\tilde{d}(3,4)$. Finally, $\tilde{d}(3,4) = \tilde{d}(2,5)$, as both pairs belong to the same class according to the *class* property and as illustrated in Table 1. Overall, we have $\tilde{d}(2,5) = 21 - \frac{1}{2}(21 - \frac{1}{2}\tilde{d}(2,5))$, i.e. $\tilde{d}(2,5) = 14$.

To generalize this example to any pair of columns, the proof exploits the following observation: our algorithm guarantees that $a[i,1] = i$ and that $a[i,2] = 2i$ if $2i \leq n$ or $a[i,2] = 2n+1-2i$ otherwise. Since A_n is symmetric, it holds:

$$\forall 1 \leq i \leq n,\ \sigma_2(i) = \begin{cases} 2i & \text{if } 2i \leq n \\ 2n+1-2i & \text{otherwise} \end{cases}$$

Without loss of generality, we consider an arbitrary pair $p_1 = (j_1, j_2)$ where $j_1 > j_2$. For the sake of completeness, we need to consider the four following cases:

- Case 1. $j_1 + j_2 \leq n$, $2j_1 \leq n$

$$\begin{aligned}
\tilde{d}(j_1,j_2) &= \Sigma min(j_1 - j_2, j_1 + j_2) && \textit{(min rule)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\tilde{d}(j_1 - j_2, j_1 + j_2) && \textit{(dist-min relation)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\Sigma min(2j_2, 2j_1) && \textit{(min rule)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\left(\frac{n(n+1)}{2} - \frac{1}{2}\tilde{d}(2j_2, 2j_1)\right) && \textit{(dist-min relation)} \\
&= \frac{n(n+1)}{4} + \frac{1}{4}\tilde{d}(\sigma_2(j_2), \sigma_2(j_1)) && \textit{(as } 2j_2 < 2j_1 \leq n\textit{)} \\
&= \frac{n(n+1)}{4} + \frac{1}{4}\tilde{d}(j_2, j_1) && \textit{(class property)}
\end{aligned}$$

- Case 2. $j_1 + j_2 \leq n$, $2j_1 > n$

$$\begin{aligned}
\tilde{d}(j_1,j_2) &= \Sigma min(j_1 - j_2, j_1 + j_2) && \textit{(min rule)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\tilde{d}(j_1 - j_2, j_1 + j_2) && \textit{(dist-min relation)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\Sigma min(2j_2, 2n+1-2j_1) && \textit{(min rule)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\left(\frac{n(n+1)}{2} - \frac{1}{2}\tilde{d}(2j_2, 2n+1-2j_1)\right) && \textit{(dist-min relation)} \\
&= \frac{n(n+1)}{4} + \frac{1}{4}\tilde{d}(\sigma_2(j_2), \sigma_2(j_1)) && \textit{(as } 2j_1 > n \textit{ and } 2j_2 \leq n\textit{)} \\
&= \frac{n(n+1)}{4} + \frac{1}{4}d(j_2, j_1) && \textit{(class property)}
\end{aligned}$$

- Case 3. $j_1 + j_2 > n$, $2j_2 > n$

$$\begin{aligned}
\tilde{d}(j_1, j_2) &= \Sigma min(j_1 - j_2, 2n + 1 - j_1 - j_2) && \textit{(min rule)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\tilde{d}(j_1 - j_2, 2n + 1 - j_1 - j_2) && \textit{(dist-min relation)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\Sigma min(2n + 1 - 2j_1, 2n + 1 - 2j_2) && \textit{(min rule)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\left(\frac{n(n+1)}{2} - \frac{1}{2}\tilde{d}(2n + 1 - 2j_1, 2n + 1 - 2j_2)\right) && \textit{(dist-min relation)} \\
&= \frac{n(n+1)}{4} + \frac{1}{4}\tilde{d}(\sigma_2(j_1), \sigma_2(j_2)) && \textit{(as } 2j_1 > 2j_2 > n\textit{)} \\
&= \frac{n(n+1)}{4} + \frac{1}{4}\tilde{d}(j_1, j_2) && \textit{(class property)}
\end{aligned}$$

- Case 4. $j_1 + j_2 > n$, $2j_2 \leq n$

$$\begin{aligned}
\tilde{d}(j_1, j_2) &= \Sigma min(j_1 - j_2, 2n + 1 - j_1 - j_2) && \textit{(min rule)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\tilde{d}(j_1 - j_2, 2n + 1 - j_1 - j_2) && \textit{(dist-min relation)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\Sigma min(2n + 1 - j_1, 2j_2) && \textit{(min rule)} \\
&= \frac{n(n+1)}{2} - \frac{1}{2}\left(\frac{n(n+1)}{2} - \frac{1}{2}\tilde{d}(2n + 1 - j_1, 2j_2)\right) && \textit{(dist-min relation)} \\
&= \frac{n(n+1)}{4} + \frac{1}{4}\tilde{d}(\sigma_2(j_2), \sigma_2(j_1)) && \textit{(as } 2j_1 > n \textit{ and } 2j_2 \leq n\textit{)} \\
&= \frac{n(n+1)}{4} + \frac{1}{4}d(j_1, j_2) && \textit{(class property)}
\end{aligned}$$

In any of these four cases, we have $\tilde{d}(j_1, j_2) = \frac{n(n+1)}{4} + \frac{1}{4}\tilde{d}(j_1, j_2)$. Therefore $\tilde{d}(j_1, j_2) = \frac{n(n+1)}{3}$ for any pair (j_1, j_2). As a result, $\tilde{A}_n$ is a balanced square. Consequently, A_n is also balanced, since a reordering of the rows does not affect the balancedness property. □

We have proved that A_n is indeed a SBLS for $2n+1$ prime. Finally, one may wonder how many orders of SBLSs we can actually generate, compared to all possible orders. To answer this, we notice that $2n + 1$ prime implies $n \neq 3k + 1$ (as otherwise, $2n + 1 = 6k + 3 = 0 \pmod 3$). Furthermore, the *prime number theorem* states that the number of primes less than x is approximately $x/ln(x)$. Hence, the density of primes among the odd numbers not divisible by 3 and lower than $2n + 1$ is approximately $3/ln(2n + 1)$. Empirically, this translates into a proportion of orders of SBLSs generated with our algorithm equal to 72% of all possible orders less than 50, 66% of the ones less than 100, and 50% of the ones less than 500.

The following section shows the results that we obtain with our algorithm.

5 Experimental Results

We compare our approach to those mentioned in Section 3. Table 3 gathers the results of all approaches. `CP` corresponds to the standard constraint programming, while `CPSS` and `CPCS` represent the constraint programming approaches based on self-symmetry and composition, respectively. The results for these three approaches are reported in [12]. `LSS` is the local search streamlining that was proposed in [13]. Finally, `U&B` corresponds to the construction that we propose in this paper.

Table 3: Solution times (in seconds) of the different approaches. A blank means that no SBLS could be generated for the corresponding order.

Order	CP	CPSS	CPCS	LSS	U&B
6	0.06	0.05	0.02	0.00	0.00
8	16.00	0.88		0.00	0.00
9	241.00	0.91		0.00	0.00
11		9.84		0.00	0.00
12		531.00	14.40	0.00	
14		5,434.00		0.02	0.00
15				0.01	0.00
17				0.25	
18			107,000.00	2.30	0.00
20				16.00	0.00
21				16.00	0.00
23				104.00	0.00
24				281.00	
26				609.00	0.00
27				4,000.00	
29				23,000.00	0.00
30				160,000.00	0.00
32				1,200,000.00	
33				1,200,000.00	0.00
35				1,200,000.00	0.00
36					0.00
39					0.00
41					0.00
44					0.00
48					0.00
50					0.00
51					0.00
53					0.00
54					0.00
56					0.00
...					...
999					0.02

6 Conclusion

In this paper we propose a polynomial time algorithm to generate spatially balanced *Latin* squares of any order n where $2n + 1$ is prime. Latin squares are central to experimental design [14, 15, e.g.], and [14]. SBLSs are particularly suited to minimize overall correlations within experiments. Experimentally, we were able to generate a SBLS of order 999 in a fraction of a second, while the largest ever known SBLS was of order 35. This work confirms a conjecture that was raised in [12].

Nonetheless, interesting research questions regarding these particular combinatorial designs remain open. First, we naturally wonder whether there is a construction for SBLSs of any order n, regardless of the primality of $2n+1$. Second, we would like to characterize additional properties and singularities of the square generated with our algorithm. Finally, determining whether the partial SBLS completion problem is $\mathcal{NP}$-complete remains an open question.

Acknowledgments

This work was supported by the National Science Foundation (NSF Expeditions in Computing award for Computational Sustainability, grant 0832782; NSF IIS award, grant 0514429) and the Intelligent Information Systems Institute, Cornell University (Air Force Office of Scientific Research, AFOSR, grant FA9550-04-1-0151). The first author was partially funded by an NSERC fellowship.

References

[1] L. Euler, Recherches sur un nouvelle espèce de quarrés magiques, Verhandelingen uitgegeven door het zeeuwsch Genootschap der Wetenschappen te Vlissingen 9 (1782) 85–239.

[2] R. C. Bose, S. S. Shrikhande, On the falsity of euler's conjecture about the non-existence of two orthogonal latin squares of order 4t+2, Proc. Nat. Acad. of Sc. U.S.A.

[3] H. van Es, C. van Es., The spatial nature of randomization and its effects on field experiments, Agron. J. 85 (1993) 420–428.

[4] H. M. V. Es, C. P. Gomes, M. Sellmann, C. L. V. Es, Spatially-balanced complete block designs for field experiments, Geoderma Journal 140 (2007) 346–352.

[5] K. Heinrich, Perfect latin squares, Discrete Applied Mathematics 37-38 (1) (1992) 281–286.
URL `http://dx.doi.org/10.1016/0166-218X(92)90139-2`

[6] C. J. Colbourn, J. H. Dinitz, Mutually Orthogonal Latin Squares: A Brief Survey of Constructions (1999).
URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.7739`

[7] L. Zhu, Completing the spectrum of r-orthogonal Latin squares, Discrete Mathematics 268 (1-3) (2003) 343–349.
URL `http://dx.doi.org/10.1016/S0012-365X(03)00053-0`

[8] C. Colbourn, Embedding partial steiner triple systems is NP-complete, Journal of Combinatorial Theory, Series A 35 (1) (1983) 100–105.
URL `http://dx.doi.org/10.1016/0097-3165(83)90031-6`

[9] C. Colbourn, The complexity of completing partial Latin squares, Discrete Applied Mathematics 8 (1) (1984) 25–30.
URL `http://dx.doi.org/10.1016/0166-218X(84)90075-1`

[10] D. Bryant, Completing partial commutative quasigroups constructed from partial Steiner triple systems is NP-complete, Discrete Mathematics 309 (14) (2009) 4700–4704.
URL `http://dx.doi.org/10.1016/j.disc.2008.05.037`

[11] C. Gomes, M. Sellmann, C. V. Es, H. V. Es, The challenge of generating spatially balanced scientific experiment designs, in: In CP-AI-OR04, 2004, pp. 387–394.

[12] C. P. Gomes, M. Sellmann, Streamlined constraint reasoning, in: M. Wallace (Ed.), CP, Vol. 3258 of Lecture Notes in Computer Science, Springer, 2004, pp. 274–289.

[13] C. Smith, C. Gomes, C. Fernandez, Streamlining local search for spatially balanced latin squares, in: IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005, pp. 1539–1541.

[14] W. G. Cochran, G. M. Cox, Experimental designs, Oxford, England: John Wiley and Sons., 1957.

[15] J. L. Fleiss, Front Matter, John Wiley and Sons, Inc., 1999, pp. i–xiv.
URL `http://dx.doi.org/10.1002/9781118032923.fmatter`