Small Latin arrays have a near transversal

Darcy Best | Kyle Pula | Ian M. Wanless

School of Mathematics, Monash University, Australia

Correspondence
Ian M. Wanless, School of Mathematics, Monash University, Clayton, VIC 3800, Australia.
Email: ian.wanless@monash.edu

Funding information
Natural Sciences and Engineering Research Council of Canada; Australian Research Council,
Grant/Award Number: DP150100506

Abstract
A Latin array is a matrix of symbols in which no symbol occurs more than once within a row or within a column. A diagonal of an $n \times n$ array is a selection of $n$ cells taken from different rows and columns of the array. The weight of a diagonal is the number of different symbols on it. We show via computation that every Latin array of order $n \leq 11$ has a diagonal of weight at least $n - 1$. A corollary is the existence of near transversals in Latin squares of these orders. More generally, for all $k \leq 20$ we compute a lower bound on the order of any Latin array that does not have a diagonal of weight at least $n - k$.

KEYWORDS
Brualdi’s conjecture, Latin square, Latin array, near transversal, partial transversal

MATHEMATICAL SUBJECT CLASSIFICATION
05B15

1 | INTRODUCTION

In this article, we examine several different types of arrays of symbols. An $m \times n$ array is row-Latin (resp., column-Latin) if each row (resp., column) contains each symbol at most once. If the array is both row-Latin and column-Latin, then we say that the array is Latin. An equi-$n$-square is an $n \times n$ array where each symbol is represented exactly $n$ times, with no row or column restrictions. A Latin square is a Latin equi-$n$-square.

A diagonal of an $n \times n$ array is a selection of $n$ cells from different rows and columns of the array. The weight of a diagonal is the number of different symbols on it. An entry in an array is a triple $(r, c, s)$ where $s$ is the symbol in cell $(r, c)$ of the array. A partial transversal of length $\ell$ is a set of $\ell$ entries, each selected from different rows and columns of a matrix, such that no two of the entries contain the same symbol. In an $m \times n$ matrix, partial transversals of length
min(m, n) and min(m, n) − 1, respectively, are known as transversals and near transversals. Hence, in an n × n array, a diagonal of weight n is a transversal and a diagonal of weight n − 1 contains two near transversals. Note that in our terminology, partial transversals are defined for non-square arrays, but diagonals are not.

It is known that there are a vast number of Latin squares that do not contain a transversal [10]. Thus, the best we can hope for in general is a near transversal. The following conjecture has been attributed to Brualdi (see [11, p. 103]) and Stein [25] and, in [13], to Ryser. It has recently been proved for Cayley tables of finite groups [17]. For several generalisations of the conjecture in terms of hypergraphs, see [1] (although some of those generalisations have since been shown to fail [16]).

**Conjecture 1.** Every Latin square contains a near transversal.

The main result of this paper is that every Latin array of order n ≤ 11 has a diagonal of weight at least n − 1. In particular, Conjecture 1 holds for Latin squares of order n ≤ 11. Our method will be computational, but it is worth bearing in mind that the number of Latin squares of orders up to 11 is too large to treat them individually [20], and the number of Latin arrays of these orders is presumably many orders of magnitude larger. Hence the key to the viability of our computation is to eliminate candidate counterexamples on the basis of limited partial information about their structure. In particular, we will need to consider partial arrays, where some cells may be empty (contain no symbol). Empty cells cannot be chosen in a partial transversal, but may be included in a diagonal (in which case, they do not contribute to the weight of that diagonal).

Since the result we are proving for small squares is more general than Conjecture 1, we now review earlier attempts to broaden that conjecture in various directions. In 1975, Stein [25] studied transversals (under the name “Latin transversals”) in equi-n-squares. He made the following seven interrelated conjectures:

**Conjecture 2.**

1. Every equi-n-square has a near transversal.
2. Every n × n array in which no symbol appears more than n − 1 times has a transversal.
3. Every (n − 1) × n array in which no symbol appears more than n times has a transversal.
4. Every (n − 1) × n row-Latin array has a transversal.
5. Every m × n array (where m < n) in which no symbol appears more than n times has a transversal.
6. Every (n − 1) × n array in which each symbol appears exactly n times has a transversal.
7. Every m × n array (where m < n) in which no symbol appears more than m + 1 times has a transversal.

Note that modern authors often include a requirement that for an array to be row-Latin it should have the same symbols in each row. However, Stein did not include that restriction, so in this paper we do not either.

Stein’s conjectures are closely related to one another (with some being special cases of others). Unfortunately, all but one of these conjectures has been disproven (only number (4) remains open). In 1998, Drisko [12] gave the (transpose of) the following construction. The
proof that we give is new, and in the spirit of the Delta Lemma (see [27]). Here and henceforth, row and column indices always start at 0.

**Theorem 3.** Let \( m \) and \( n \) be integers satisfying \( m < n \leq 2m - 2 \). Define an \( m \times n \) column-Latin array \( A = [a_{ij}] \) on symbols \( \{0, \ldots, m-1\} \), by

\[
a_{ij} \equiv \begin{cases} 
i \mod m & \text{if } j \in \{0, \ldots, m-2\}, \\i + 1 \mod m & \text{if } j \in \{m-1, \ldots, n-1\}. 
\end{cases}
\]

Then \( A \) has no transversals.

**Proof.** Define \( \Delta(i,j) = a_{ij} - i \) for \( 0 \leq i < m \) and \( 0 \leq j < n \). Suppose that \( A \) has a transversal on cells \((0,j_0), \ldots, (m-1,j_{m-1})\). Then since every symbol in \( \{0, \ldots, m-1\} \) appears in the transversal it follows that

\[
\sum_{i=0}^{m-1} \Delta(i,j_i) \equiv \sum_{i=0}^{m-1} i - \sum_{i=0}^{m-1} i \equiv 0 \mod m.
\]

However, this congruence is impossible to satisfy, given that \( \Delta(i,j) = 0 \) for \( j \leq m-2 \) and \( \Delta(i,j) \equiv 1 \mod m \) for \( j > m-2 \), so that

\[
\sum_{i=0}^{m-1} \Delta(i,j_i) \equiv |\{i : j_i > m-2\}| \neq 0 \mod m.
\]

The \( m = n - 1 \) case of Theorem 3 is a direct counterexample to Conjecture 2 parts (3), (5), (6) and (7). Also Stein and Szabó [26] gave counterexamples to part (2) when \( n \in \{5, 6\} \). Furthermore, Pokrovskiy and Sudakov [23] recently gave a constructive proof of the following.

**Theorem 4.** For all sufficiently large \( n \) there exists an equi-\( n \)-square that does not have a partial transversal of length more than \( n - \frac{1}{42}\log n \).

This is a counterexample to Conjecture 2(1). Moreover, Pokrovskiy and Sudakov showed how their result can be extended to give further counterexamples to statements like Conjecture 2(2) as well. For some \( n > e^{84} \), use Theorem 4 to construct an equi-\( n \)-square \( E \) of order \( n \) with no partial transversal of length \( n - 2 \). Now, form a matrix \( A \) of order \( n + 1 \) by adding one row and one column to \( E \), where the new row and column contain \( 2n + 1 \) distinct symbols that do not appear in \( E \). Since at most two of these new symbols can be used in any partial transversal, \( A \) does not have a transversal. By a similar argument, we can pad an appropriate equi-\( n \)-square with either new symbols or the original symbols to provide counterexamples to parts (3), (5), (6) and (7) that are of a different nature to the counterexamples provided by Theorem 3. Interestingly, [22] shows that almost all equi-\( n \)-squares have a transversal, so these counterexamples may be viewed as atypical.

The only one of Stein’s conjectures that remains unsolved is (4). While we are quite unsure about this conjecture, it seems much more promising if we also enforce the array to be column-Latin.
Conjecture 5. Let $R$ be an $(n - 1) \times n$ Latin array. Then $R$ contains a transversal.

Note that this conjecture implies a strengthened form of Conjecture 1, where we may choose which row (or column or symbol) is not included in our partial transversal. It is an open question (see [4]) whether an even stronger property holds when $n$ is large: it may be that all large $(n - 1) \times n$ Latin arrays on $n$ symbols have a decomposition into transversals. Often such arrays are called Latin rectangles. Interestingly, it is not possible to relax the requirement on the number of symbols. We know there are many Latin squares without transversals [10]. If a column of previously unused symbols is appended to such a Latin square we create a Latin array with no decomposition into transversals. Alternatively, if we remove the first two rows and the first column from the Cayley table of an elementary abelian group of order $2^k > 2$, then we get a $(2^k - 2) \times (2^k - 1)$ Latin array on $2^k$ symbols, with no decomposition into transversals. This follows from a result of Akbari and Alireza [3]. Nevertheless, Latin arrays that are in some sense far from being Latin squares are known to have many transversals. Montgomery et al. [21] recently showed that an $n \times n$ Latin array in which at most $(1 - o(1))n$ symbols occur more than $(1 - o(1))n$ times has $(1 - o(1))n$ pairwise disjoint transversals.

A partial transversal is maximal if it is not contained in any longer partial transversal. It is not hard to see that a maximal partial transversal of length $\ell$ in a Latin square of order $n$ must satisfy $n/2 \leq \ell \leq n$. In [6] it was shown that for $n \geq 5$ all values of $\ell$ in this range are achieved. Then Evans [14] constructed an infinite family of Latin squares which simultaneously have maximal partial transversals of each of the permissible lengths. Subsequently, Evans et al. [15] showed that there exists a Latin square of order $n$ which has maximal partial transversals of each permissible length if and only if $n \not\in \{3, 4\}$ and $n \not\equiv 2 \pmod{4}$.

In Latin squares, it is easy to find a partial transversal of length $\lfloor n/2 \rfloor$ using a greedy algorithm. A succession of results have progressively improved on this observation; see [27] for a history. The three most important breakthroughs have been the following.

In 1978, Brouwer et al. [8] and Woolbright [28] independently achieved the first lower bound that is asymptotically equal to $n$.

Theorem 6. Every Latin square of order $n$ contains a partial transversal of length at least $n - \sqrt{n}$.

Correcting an earlier flawed proof, in 2008 Shor and Hatami [18] improved the deficit to $O(\log^2 n)$.

Theorem 7. Every Latin square of order $n$ contains a partial transversal of length at least $n - 11.053 \log^2 n$.

Finally, very recently Keevash et al. [19] improved the error term once again.

Theorem 8. Every Latin square of order $n$ contains a partial transversal of length at least $n - O(\log n / \log \log n)$.

The goal of this paper is to provide, for $k \leq 20$, an improved lower bound on the order $n$ of any Latin array that lacks a partial transversal of length $n - k$. The case $k = 1$ is handled in the next section, and the case $2 \leq k \leq 20$ is addressed in the last section. Our results rely
heavily on computation. Each computation was verified by at least two independent programmes. Preliminary versions of our results were given in the PhD theses of the first two authors [5,24].

2 | NEAR TRANSVERSALS

In this section we describe our approach to proving that Latin arrays of order $n \leq 11$ have a near transversal. Our method is based on the work of Shor and Hatami [18]. Until the recent breakthrough by Keevash et al. [19], Theorem 7 was the state of the art. The key idea needed to prove it was the idea of $\#$-swapping. Consider a diagonal, $T$, of weight $w$. Choose two entries from $T$, say $(i_0, j_0, k_0)$ and $(i_1, j_1, k_1)$. If $T \backslash \{(i_0, j_0, k_0), (i_1, j_1, k_1)\}$ still covers $w$ symbols, then we consider the diagonal

$$(T \backslash \{(i_0, j_0, k_0), (i_1, j_1, k_1)\}) \cup \{(i_0, j_1, \cdot), (i_1, j_0, \cdot)\},$$

where we adopt the convention of using $\cdot$ to denote an unknown symbol (possibly a different symbol each time the notation is used). This diagonal is guaranteed to have a weight of $w, w + 1$ or $w + 2$. The act of swapping $\{(i_0, j_0, k_0), (i_1, j_1, k_1)\}$ to obtain a new diagonal is called a $\#$-swap. Note that by repeated use of $\#$-swaps, the weight of the diagonal can never decrease. Thus, if we start with a diagonal of maximum weight, it is impossible to $\#$-swap to a diagonal of larger weight and the set of symbols on each diagonal that we reach by $\#$-swapping will be the same.

Throughout the remainder of the section, we use the symbol $\times$ to indicate a cell that must contain a symbol that appears on the original diagonal. For example, if that cell is reachable via a sequence of $\#$-swaps and the original diagonal has maximum weight, then the symbol in the cell must appear somewhere on the original diagonal. In contrast, if a cell is shown as empty, it means that we do not know anything about it.

**Example 9.** Here is an example of $\#$-swapping on a diagonal of weight $4 = 6 - 2$. If we remove the top left 0 and 1 from the diagonal, we still have 4 symbols left, so we may $\#$-swap on these entries and instead consider the diagonal which contains the two $\times$’s, with the bottom four rows unchanged.

Note that after performing a $\#$-swap, the two cells that were swapped out will be lightly shaded for further clarity.

The primary purpose of this paper is to describe a proof of a generalisation of Conjecture 1 for small orders. We try to find near transversals in all Latin arrays rather than just Latin
squares. We focus on diagonals of weight $n - 2$ and attempt to uncover a new symbol, which would locate a near transversal. The following elementary observation is needed throughout.

**Lemma 10.** If every Latin array of order $n$ contains a partial transversal of length $k$, then every Latin array of order $n + 1$ contains a partial transversal of length $k$.

Throughout the section, we utilise Lemma 10 iteratively. Having shown that all Latin arrays of order $n - 1$ contain a near transversal, we will then know that all Latin arrays of order $n$ contain a diagonal of weight at least $n - 2$. A diagonal of weight $n - 2$ has two essentially different configurations for the duplicated symbols as shown in the following pictures:

![Type A and Type B diagrams](image)

A diagonal of type A has two symbols which each occur twice on the diagonal, whilst a diagonal of type B has one symbol that occurs three times on the diagonal. We first start by showing that the existence of a type B diagonal implies the existence of a type A diagonal in maximal cases.

**Lemma 11.** Let $L$ be a Latin array of order $n$ with a diagonal of weight $n - 2$ and no diagonal of weight greater than $n - 2$. If $L$ has a diagonal of type B, then there exists a diagonal of type A that can be reached by a sequence of #-swaps.

**Proof:** Assume, on the contrary, that no diagonal of type A can be reached. Without loss of generality, the initial diagonal of type B is the main diagonal and the three repeated symbols are in the top three rows.

We need to perform $n - 2$ #-swaps to arrive at a contradiction. First, we #-swap $(0, 0, 0)$ and $(2, 2, 0)$. The symbols in the cells $(0, 2)$ and $(2, 0)$ must be the same, otherwise this new diagonal would be of type A. Without loss of generality, these cells contain the symbol 1. We now #-swap $(0, 2, 1)$ and $(3, 3, 1)$. By a similar argument, the
two uncovered cells must contain the same symbols (which is, without loss of generality, 2). We repeat this same argument \( n - 2 \) times in total. On all steps \( i \) (except the first one), we #-swap the entries \((0, i, i - 1)\) and \((i + 1, i + 1, i - 1)\) and expose the entries \((0, i + 1, i)\) and \((i + 1, i, i)\). The first three steps are shown in Figure 1.

However, at step \( n - 2 \), the uncovered symbol must be some symbol that did not appear on the original diagonal. Thus, we have found a heavier diagonal, a contradiction. \( \square \)

Our next result generalises Lemma 11, except that we abandon the condition that we must be able to #-swap to the new diagonal. It also generalises [9, Prop.7], whose proof it mimics. Note that [9, Prop.7] has been generalised in a different direction (namely, to row-Latin arrays of order \( n \) containing \( n \) symbols) by Aharoni et al. [2].

**Lemma 12.** Any entry of a Latin array contained in a diagonal of weight \( w \) is contained in a diagonal of weight at least \( w \) where each symbol appears on the diagonal at most twice.

**Proof.** Let \( L \) be a Latin array of order \( n \). For convenience, we will assume that the diagonal in question is the main diagonal and let \( M \) be the multiset of symbols on the main diagonal. If no symbol appears more than twice in \( M \), we are done. Otherwise, fix some entry \((r, r, \cdot)\). We will find a diagonal of weight at least \( w \) with the desired properties that still contains \((r, r, \cdot)\).

Select \( n \neq r \) such that the symbol \( L(r_1, r_1) \) appears in \( M \) three or more times. Let \( x_i \) be the number of symbols appearing exactly \( i \) times in \( M \). It follows that

\[
\sum_{i=0}^{n} ix_i = n.
\]

Thus, \( n = (x_1 + x_2 + \cdots + x_n) = 2x_2 + 2x_3 + \cdots + (n - 1)x_{n-1} + x_n \).

Row \( r_1 \) contains at least \( n - (x_1 + x_2 + \cdots + x_n) = x_2 + 2x_3 + \cdots + (n - 1)x_{n-1} + x_n \) symbols that do not appear in \( M \) and column \( r_1 \) contains \( \left(\sum_{i=2}^{n} x_i\right) - 1 \) symbols that appear more than once in \( M \), besides the entry \((r_1, r_1, \cdot)\). Thus, there are at least two values of \( r_2 \) such that the symbol \( L(r_1, r_2) \) does not appear in \( M \) and the symbol \( L(r_2, r_1) \) does not appear more than once in \( M \). Select \( r_2 \neq r \). Observe that

\[
T = (M \setminus \{(r_1, r_1, \cdot), (r_2, r_2, \cdot)\}) \cup \{(r_1, r_2, \cdot), (r_2, r_1, \cdot)\}
\]
has fewer cells than $M$ that contain symbols that appear more than twice in the diagonal and $(r, r, \cdot)$ still belongs to $T$. Furthermore, $T$ is of weight at least $w$. By iterating, we will therefore find a diagonal with the desired properties.

Next, we give a simple example of how using #-swaps is useful.

**Lemma 13.** In any Latin array of order 6, there exists a diagonal of weight at least 5.

**Proof.** First, it is quite easy to show that the heaviest diagonal must be at least of weight 4 (e.g., use Theorem 6). We now assume, on the contrary, that there exists a Latin array that contains a diagonal of weight 4, but none of weight 5 or 6. Without loss of generality, the original diagonal of length 4 is along the main diagonal. By Lemma 11, we may assume that it takes the form given here.

At this point, we are presented with four options for which pair of entries to #-swap (choose either 0 and either 1 independently). From this, we can see that we have the following.

As explained above, each $\times$ must be one of 0, 1, 2, 3; otherwise, we would have a heavier diagonal. Consider #-swapping the entries in the first row and the third row. The symbol in the $(2, 0)$ cell must be either 2 or 3. Without loss of generality, we assume that it is a 2.
Note that we do not know what symbol is in the \((0, 2)\) cell, but we do know that it is a duplicate symbol (i.e., it appears at least one more time on the diagonal or at least two more times if it is a 2). Thus, we are free to \#-swap on that entry now. We \#-swap that entry and \((4, 4, 2)\).

\[
\begin{array}{ccc}
0 & \times & \times \\
\times & \times & \\
2 & 1 & \\
\times & \times & 1 \\
& \times & 2 \\
& & 3
\end{array}
\]

The symbol in the \((4, 2)\) cell must be either 0 or 3 and the symbol in the \((0, 4)\) cell is a duplicate (as described above), and so may be used immediately. At this point, we consider both cases for the \((4, 2)\) cell separately. In either case, we \#-swap the entry in the top row with the appropriate duplicated symbol.

\[
\begin{array}{ccc}
0 & \times & \times \\
\times & \times & \\
2 & 1 & \\
\times & \times & 1 \\
0 & 2 & \\
& 3
\end{array}
\quad
\begin{array}{ccc}
0 & \times & \times \\
\times & \times & \\
2 & 1 & \\
\times & \times & 1 \\
3 & 2 & \\
& \times
\end{array}
\]

In either case, the top row now has five entries whose symbol must come from the set \(\{0, 1, 2, 3\}\), which is impossible in a Latin array. The result follows.

Hatami and Shor [18] used this same idea to show Lemma 13. However, in their description, they did not leave all of the symbols in the top row as unknown (\(\times\)). Instead, they did extra case analysis to determine what those symbols could be. By leaving the top row as unknown symbols, there is the potential for less branching in the algorithm. Moreover, by continually using the top row to \#-swap on, there are only two choices of pairs of entries to \#-swap (but one of these choices undoes the last change and reverts to the previous diagonal).

We describe two algorithms whose goal is to show that there is a near transversal in all Latin arrays of a particular order \(n\). Algorithm 1 describes the basic algorithm to show that all Latin arrays of order \(n\) contain a near transversal. This algorithm formalises the method used in the proof of Lemma 13. This algorithm is then refined in Algorithm 2, which succeeds for larger orders than Algorithm 1 does.

It is important to note that in both algorithms below, all variables are considered local variables, so changing the value of a parameter does not affect its value outside of that specific instance.
Algorithm 1 Basic algorithm to show that all Latin arrays of order \(n\) contain a near transversal.

\[ \text{Algorithm 1} \]

**Input** \(L\) is a partial Latin array with some otherwise empty cells marked with \(\times\)

**Input** \(\sigma\) is a permutation defining a diagonal of weight \(n - 2\) in \(L\)

**Input** \(d\) is the depth of the search

**Input** \(r\) is the row we just hashed on

**Output** \(\text{TRUE}\) if every Latin array that is a completion of the input has a near transversal.

**Output** \(\text{FALSE}\) if the computation is inconclusive.

1: procedure \(\text{NaiveHash}(L, \sigma, d, r)\)
2: if Some row or column of \(L\) contains at least \(n - 1\) filled cells then
3: return \(\text{TRUE}\) \(\triangleright\) Near transversal guaranteed
4: if \(d \neq 0\) and \(\sigma\) is the identity and \(r = 3\) then
5: return \(\text{FALSE}\) \(\triangleright\) We have cycled back to where we started
6: \(S \leftarrow L(r, \sigma_r)\) \(\triangleright\) Symbol to hash on
7: \(R \leftarrow \text{row such that } \sigma_R = S, R > 0 \text{ and } R \neq r\) \(\triangleright\) Other row that contains \(S\) on \(\sigma\)
8: \(\text{swap}(\sigma_0, \sigma_R)\) \(\triangleright\) Update \(\sigma\) to enact the \#-swap
9: fill cells \((0, \sigma_0)\) and \((R, \sigma_R)\) \(\triangleright\) Set to \(\times\) if they do not already contain a symbol
10: if \(L(R, \sigma_R) \neq \times\) then
11: return \(\text{NaiveHash}(L, \sigma, d + 1, R)\) \(\triangleright\) If we already know what symbol this is
12: else
13: \(k \leftarrow \text{largest symbol in } L\) that appears multiple times
14: \(\triangleright\) The symbols \(k + 1, k + 2, \ldots, n - 3\) are all symmetric up to this point, so we only need to consider one of them
15: \(\triangleright\) (without loss of generality, we use \(k + 1\)).
16: \(s \leftarrow 0\) to \(\min(k + 1, n - 3)\) do
17: \(\text{if } s\text{ is not in row } R\text{ nor column } \sigma_R\text{ then}\)
18: \(L(R, \sigma_R) \leftarrow s\)
19: \(\text{if } \text{NaiveHash}(L, \sigma, d + 1, R) = \text{FALSE}\text{ then}\)
20: return \(\text{FALSE}\)
21: return \(\text{TRUE}\) \(\triangleright\) No matter which symbol we use, there is a near transversal

Algorithm 1 is sufficient to show that all Latin arrays of order \(n \leq 7\) contain a near transversal (this result was obtained by an independent calculation in [7]). However, for \(n = 8\), Algorithm 1 fails to show the desired result as it returns False for Figure 2.

A total of 14 partial Latin arrays fail Algorithm 1 for \(n = 8\) (meaning that Line 5 of Algorithm 1 is reached 14 times). For \(n = 9\), one may expect more squares to fail Algorithm 1, but interestingly, those 14 squares (with one extra row and column added) are the only squares to fail Algorithm 1. For \(n = 10\), a total of 82 140 squares fail Algorithm 1. By comparison, Line 3 of Algorithm 1 was reached 2 657, 377 452 and 696 808 457 times, respectively, for \(n = 8, 9, 10\).

The failures such as Figure 2 show that a more refined approach is needed to find near transversals in larger orders. The first observation is that after we have cycled back on ourselves and returned False on Line 5 of Algorithm 1, we may now choose another row to \#-swap on, rather than the first one. Recall that by only using \#-swaps on the top row, we are only utilising
two of the possible \#-swaps available (there are 4 possible if the diagonal is of type A and 3 if it is of type B). We also note in passing that, although our searches are always based around the main diagonal, there are other options available as soon as other diagonals of weight $n - 2$ have been located.

In Algorithm 2, we first \#-swap along the top row. Once we cycle around, we then \#-swap along the second row, then the third, then the fourth. In Algorithm 1, we arbitrarily selected row 3 to be the initial value for $r$. In Algorithm 2, when we are \#-swapping on rows 0, 1, 2 and 3, we use the rows 3, 2, 1 and 0, respectively for the initial value of the “row we just \#-swapped on”. It was convenient, but far from essential, to know that $r_0 + r_1 = 3$.

There are two heuristics that can be added to the search that significantly improve its performance when utilised together. (However, there is a minor drawback to using them, which we will discuss in Section 3). The first heuristic is to search for diagonals of weight $n - 2$ that may not be reachable via \#-swaps. If a partial transversal of length $n - 2$ covers all rows except $r_0$ and $r_1$ and all columns except $c_0$ and $c_1$, then we know that each of the cells $(r_0, c_0), (r_0, c_1), (r_1, c_0)$ and $(r_1, c_1)$ must also contain symbols from $\{0, 1, ..., n - 3\}$, or else we would have a near transversal. Thus, if those cells are empty, we may fill them with an $\times$. In practice, every time that we fill a cell with a specific symbol (not an $\times$), we only search for diagonals that go through that cell. The second heuristic is to choose some $\times$ in the square and decide what that symbol should be by exhaustively trying each one. We define the liberties of a cell to be the number of symbols that could be placed into the cell without violating the Latin property. In practice, we choose an $\times$ that has the fewest liberties to limit the branching that our search does. These heuristics are combined in Algorithm 2.
Algorithm 2 is good enough to show the following result. Both implementations of the algorithm reached Line 5 the same number of times (namely, 53 times for $n = 10$ and 105 287 times for $n = 11$ with $r_0 = 0$ in all cases) providing some corroboration of each other.

```
Algorithm 2
More advanced algorithm to determine if all Latin arrays of order $n$ contain a near transversal. Hash($L, \varepsilon, 0, 0, 3$) should be called initially, where $L$ is an $n \times n$ ($n \geq 4$) array with all cells empty except the main diagonal, which contains $\{0, 0, 1, 1, 2, 3, \ldots, n - 3\}$ and $\varepsilon$ is the identity permutation. FillCell simply tries all valid symbols to place in the cell $(r, c)$ and calls Hash with the same parameters, but with depth $d + 1$.

Input $L$ is a partial Latin array with some otherwise empty cells marked with $\times$
Input $\sigma$ is a permutation defining a diagonal of weight $n - 2$
Input $d$ is the depth of the search with the current $r_0$
Input $r_0$ is the row we are mainly $\#$-swapping on (this was the top row in Algorithm 1)
Input $r_1$ is the other row that we just $\#$-swapped on
Output True if every Latin array that is a completion of the input has a near transversal.

1: procedure Hash($L, \sigma, d, r_0, r_1$)
2:     if Some row or column of $L$ contains at least $n - 1$ filled cells then
3:         return True ▷ Near transversal guaranteed
4:     if $d \neq 0$ and $\sigma$ is the identity and $r_0 + r_1 = 3$ then ▷ We have cycled back
5:         if $r_0 \geq 3$ then return False ▷ We need to try something different.
6:         else return Hash($L, \sigma, 0, r_0 + 1, r_1 - 1$) ▷ Try $\#$-swapping along the next row.
7:     if $d \equiv 3$ (mod 4) and there is at least one $\times$ in $L$ then
8:         $(r, c) \leftarrow$ cell such that $L(r, c) = \times$. ▷ If there are multiple $\times$, select one with the
9:         return FillCell($L, r, c, \sigma, d, r_0, r_1$) ▷ fewest liberties, breaking ties by selecting
10:            the first one in row-major order.
11:     return FillCell($L, r, c, \sigma, d, r_0, r_1$)
12:  END.
13: procedure FillCell($L, r, c, \sigma, d, r_0, r_1$)
14:     if $L(r, c) \in \{0, \ldots, n - 3\}$ then return Hash($L, \sigma, d + 1, r_0, r_1$)
15:     $k \leftarrow$ largest symbol in $L$ that appears multiple times
16:     for $s \leftarrow 0$ to $\min(k + 1, n - 3)$ do
17:         if $s$ is not in row $r$ nor column $c$ then
18:             $L' \leftarrow L$ ▷ Store a copy of $L$
19:             $L(r, c) \leftarrow s$
20:             for each Partial transversal, $T$, of length $n - 2$
21:                 $\{R_1, R_2, C_1, C_2\} \leftarrow$ the two rows and two columns missing from $T$
22:                 Fill in cells $(R_1, C_1), (R_1, C_2), (R_2, C_1), (R_2, C_2)$ with $\times$ if empty
23:             if Hash($L, \sigma, d + 1, r_0, r_1$) = False then
24:                 return False ▷ No matter which symbol we place here, there is a near transversal
25:             $L \leftarrow L'$ ▷ Restore $L$ to its previous configuration
26:     return True
```
Theorem 14. Every Latin array of order \( n \leq 11 \) contains a near transversal.

Proof. The cases where \( n < 4 \) are easy to see (they also follow from Theorem 6). We iteratively use Lemma 10 and Algorithm 2 for \( n = 4, \ldots, 11 \).

In the search for \( n = 11 \), every \#-swap included one of the first two rows (i.e., \( r_0 \in \{0, 1\} \) in Algorithm 2). The search for \( n \leq 10 \) can be completed in a matter of minutes, while a few hours is needed for \( n = 11 \). Based on this progression, we initially believed \( n = 12 \) to be possible. However, after running our programme for several months with several different pruning heuristics on a small grid, we did not think that the programme would finish in a reasonable amount of time. Due to the recursive nature of the algorithm, it is difficult to accurately determine what percentage of the search space was covered over those months. Needless to say, all cases that we searched did not provide a counterexample to Conjecture 1.

If one wishes to use Algorithm 2 to find near transversals in Latin squares (rather than in all Latin arrays), then we would recommend extra heuristics be employed. For example, the partial Latin array in Figure 3 is one of many squares that loops back in Line 4 of Algorithm 2 with \( r_0 = 0 \). The search continues with \( r_0 = 1 \), however, no further search is needed if we are only concerned with Latin squares. There are only two symbols which are not on the original diagonal (9 and 10), but there are not enough empty cells left to place them into. In particular, at least \( 2(n-2) - 4 \) cells in the top \((n-2) \times (n-2)\) submatrix must be empty to fit in the two missing symbols.

The fact that all Latin arrays, and not just Latin squares, have near transversals is an encouraging sign for Conjecture 1. It is plausible that an even stronger result holds:

Conjecture 15. Every Latin array contains a near transversal.

This conjecture is implied by Conjecture 5. Of course, the fact that our arrays are Latin seems to be a very important factor in the potential truth of either conjecture. The idea used in Theorem 4 relies heavily on clumping all \( n \) of each symbol into a small submatrix. The idea cannot be easily changed to accommodate only one of each symbol per row or column.

![Figure 3](image-url)

A partial Latin array that loops back in Line 4 when \( r_0 = 0 \)
Theorem 4 was not optimised in [23]. It would be interesting to know the smallest value of $n$ where an equi-$n$-square exists that does not contain a near transversal.

3 | LONG PARTIAL TRANSVERSALS

In this final section, we improve the lower bounds on the length of the longest partial tranversal that Latin arrays of all orders $n < 1449$ are known to possess. In the proof of Theorem 7 by Shor and Hatami [18], one of the key ingredients was sets of diagonals with the same weight that were connected by a sequence of $\#$-swaps. A sequence of integers $n_k$ was discussed in detail. To connect those to our results here, $n_2$ is defined as the smallest order such that a diagonal of weight $n - 2$ cannot be $\#$-swapped to uncover a new symbol. Note that the heuristics employed in Algorithm 2 mean that we cannot use the results from Theorem 14 to show that $n_2 \geq 12$. However, we verified that $n_2 \geq 11$ utilising a similar idea to Algorithm 1 and 2.

Upon first glance, the $n - 11.053 \log^2 n$ bound shown by Shor and Hatami [18] is weaker than the $n - \sqrt{n}$ bound in Theorem 6 for small values of $n$. In fact, for $n \leq 7731462$, it is better to use the $n - \sqrt{n}$ bound. However, the groundwork laid out in the asymptotic proof in [18] can be used in a concrete way to show significantly better bounds for lower orders. The key sequence, $n_k$, is a bound on the size that a square must have before being able to $\#$-swap from a diagonal of weight $n - k$ to a heavier one. In particular, any Latin array of order $n < n_k$ contains a diagonal with weight greater than $n - k$.

The following lemma is taken from [18], except the first inequality has been strengthened as explained above.

**Lemma 16.**

\begin{align*}
    n_2 &\geq 11, \quad (1) \\
    n_k &\geq n_{k-1} + 2k \quad \text{for} \quad k > 2 \quad \text{and} \quad (2) \\
    (n_k - n_j)(2n_j + n_{k-1} - 2n_k + 2k - j) &\leq n_j(n_j - n_{j-1} - 2j) \quad \text{for} \quad 3 \leq j < k. \quad (3)
\end{align*}

Shor and Hatami used (3) to show that $k \leq 11.053 \log^2 n_k$. While this seems worse than Theorem 6 for small values, simple induction using (1) and (2) shows that $n_k \geq k^2 + k + 5$, giving a better bound than Theorem 6 for all $n$. For small values, the value of $11.053 \log^2 n$ is far from the truth.

Searching for a single sequence that satisfies inequalities (1)–(3) is quite simple. In fact, for any sequence $[n_2, ..., n_\ell]$ that satisfies (1)–(3), you may extend it by setting $n_{\ell+1} = 2n_\ell + 2(\ell + 1)$ and it will still satisfy (1)–(3). However, the true interest lies in the smallest value that $n_k$ can achieve for each $k$. Unfortunately, a naive search is not feasible for determining this value for even modest values of $k$, so heuristics are needed to trim the search space. We start by noting that we may assume that $n_2 = 11$ since if $[n_2, n_3, ..., n_k]$ satisfies (1)–(3), then $[11, n_3, ..., n_k]$ also satisfies (1)–(3). Unfortunately, this greedy nature does not generalise to the remaining parts of the sequence. To minimise $n_k$, we may need to use nonoptimal values for $n_3, ..., n_{k-1}$. For example, $n_4 = 28$ is attainable. However, to achieve $n_5 = 41$, we must use $n_4 = 31$.

In computations to find the smallest value that $n_k$ can take, the following heuristic proved very useful. Suppose that we have a current candidate for the smallest value of $n_k$, say $\tilde{n}_k$. We say that a pair $(x, y)$ is viable if there exists a sequence $[\tilde{n}_x = y, \tilde{n}_{x+1}, ..., \tilde{n}_k]$ that satisfies (1)–(3) and $\tilde{n}_k < \kappa$. Note that (3) need only be satisfied for $x < j < k$ and that we may assume that
To determine the smallest possible value for \( n_k \), we constructed sequences that satisfied (1)–(3) and only contained viable pairs. Each time we found a sequence that had a smaller value for \( n_k \), we recomputed which pairs of \((i, j)\) were viable. Note that a pair that was marked as not viable could never be marked as viable for a smaller value of \( n_k \), so it never needs to be reconsidered. If for some \( x \) there were no viable pairs \((x, y)\), then we concluded that the current candidate for the smallest value of \( n_k \) is indeed the smallest value possible. At that point we could begin the search for the smallest possible value of \( n_{k+1} \).

Formalising this approach, say that \( v(x, y) \) is true if the pair \((x, y)\) is viable and false otherwise. Let \( s_k = [\tilde{n}_{k,2}, \tilde{n}_{k,3}, ..., \tilde{n}_{k,k}] \) denote our current candidate for a sequence achieving the lowest possible value for \( n_k = \tilde{n}_{k,k} \). We start with \( s_2 = [11] \). Then for \( k = 3, 4, ..., \) we seek the sequence \( s_k \) as follows:

1. Let \( s_k \) be the sequence formed from \( s_{k-1} \) by appending the smallest value that ensures that \( s_k \) satisfies (1)–(3) when we put \( n_i = \tilde{n}_{k,i} \) for each \( i \).

| \( k \) | One sequence \([n_2, ..., n_k]\) that minimise \( n_k \) |
|--------|--------------------------------------------------|
| 2      | [11]                                             |
| 3      | [11, 17]                                         |
| 4      | [11, 17, 28]                                     |
| 5      | [11, 17, 31, 41]                                 |
| 6      | [11, 17, 28, 46, 58]                             |
| 7      | [11, 17, 28, 42, 64, 78]                         |
| 8      | [11, 17, 28, 42, 63, 90, 107]                    |
| 9      | [11, 17, 28, 46, 58, 91, 122, 140]               |
| 10     | [11, 17, 28, 42, 64, 78, 122, 157, 177]          |
| 11     | [11, 17, 28, 42, 63, 90, 107, 165, 204, 226]     |
| 12     | [11, 17, 28, 46, 58, 91, 122, 140, 216, 259, 283]|
| 13     | [11, 17, 28, 42, 64, 78, 122, 157, 177, 272, 320, 346]|
| 14     | [11, 17, 28, 42, 64, 78, 122, 157, 177, 272, 356, 408, 436]|
| 15     | [11, 17, 28, 42, 63, 90, 107, 165, 204, 226, 364, 439, 495, 525]|
| 16     | [11, 17, 28, 46, 58, 91, 122, 140, 216, 259, 283, 432, 534, 594, 626]|
| 17     | [11, 17, 28, 42, 64, 78, 122, 157, 177, 272, 320, 346, 527, 638, 702, 736]|
| 18     | [11, 17, 28, 42, 64, 78, 122, 157, 177, 272, 356, 408, 436, 662, 783, 851, 887]|
| 19     | [11, 17, 28, 42, 63, 90, 107, 165, 204, 226, 346, 439, 495, 525, 796, 933, 1005, 1043]|
| 20     | [11, 17, 28, 46, 58, 91, 122, 140, 216, 259, 283, 432, 534, 594, 626, 948, 1110, 1192, 1234]|
| 21     | [11, 17, 28, 42, 64, 78, 122, 157, 177, 272, 320, 346, 527, 638, 702, 736, 1114, 1304, 1400, 1449]|
2. For each $j$, let $v(2,j)$ be true if and only if $j = 1$ and let $v(k,j)$ be true if and only if $\tilde{n}_{k-1,k-1} + 2k \leq j < \tilde{n}_{k,k}$.

3. For $i = k - 1, k - 2, ..., 3$; for each $j$, let $v(i,j)$ be true if and only if $j \tilde{n}_{i,i}$ and $v(i+1,j')$ is true for some $j' \geq j + 2i + 2$.

4. For $i = k - 1, k - 2, ..., 2$; for each $j$ for which $v(i,j)$ is true, check that there is a sequence $[t_i = j, t_{i+1}, ..., t_k]$ such that putting $n_x = t_x$ for $i \leq x \leq k$ satisfies (1)–(3), and where $v(x,t_x)$ is true for $i \leq x \leq k$. If not, set $v(i,j)$ to false.

5. If there is some $i$ for which $v(i,j)$ is false for all $j$, then declare the present $s_k$ to be the best possible, and begin looking for the best $s_{k+1}$. Otherwise, $v(2,11)$ must be true. Replace $s_k$ by the sequence $[t_2, ..., t_k]$ that showed that $v(2,11)$ is true. Mark $v(k,j)$ to be false for $j \geq t_k$.

Go to step 4.

Using the algorithm outlined above we were able to compute the smallest values that $n_k$ can take and satisfy Lemma 16, for $k \leq 21$. The results are presented in Table 1. Note that the same algorithm is capable of computing several further values for $n_k$, with each additional value requiring a few days of computation.

Table 1 can be used to show explicit bounds on the length of a partial transversal in a Latin square. For example, Table 1 shows that $n_{21} \geq 1449$. Thus, any Latin array of order $n < 1449$ has a partial transversal of length at least $n - 20$. By comparison, Theorem 6 only implies that any Latin array of order $n < 21^2 = 441$ has a partial transversal of length at least $n - 20$.

ACKNOWLEDGEMENTS

This study was supported by Endeavour Postgraduate Scholarship, NSERC CGS-D and ARC grant DP150100506.

ORCID

Ian M. Wanless https://orcid.org/0000-0001-8085-3087

REFERENCES

1. R. Aharoni and E. Berger, *Rainbow matchings in r-partite r-graphs*, Electron. J. Combin. 16 (2009), no. 1, R119.
2. R. Aharoni, E. Berger, D. Kotlar, and R. Ziv, *Degree conditions for matchability in 3-partite hypergraphs*, J. Graph Theory. 87 (2018), 61–71.
3. S. Akbari and A. Alireza, *Transversals and multicolored matchings*, J. Combin. Des. 12 (2004), 325–332.
4. J. Barát and I. M. Wanless, *Rainbow matchings and transversals*, Australas. J. Combin. 59 (2014), 211–217.
5. D. Best, *Transversal this, transversal that*, Ph.D. Thesis, Monash University, 2018.
6. D. Best, T. Marbach, R. J. Stones, and I. M. Wanless, *Covers and partial transversals of Latin squares*, Des. Codes Cryptogr. 87 (2019), 1109–1136.
7. D. Best, K. Hendrey, I. M. Wanless, T. E. Wilson, and D. R. Wood, *Transversals in Latin arrays with many distinct symbols*, J. Combin. Des. 26 (2018), 84–96.
8. A. E. Brouwer, A. de Vries, and R. M. A. Wieringa, *A lower bound for the length of partial transversals in a Latin square*, Nieuw Archief Voor Wiskunde. 26 (1978), 330–332.
9. P. J. Cameron and I. M. Wanless, *Covering radius for sets of permutations*, Discrete Math. 293 (2005), 91–109.
10. N. J. Cavenagh and I. M. Wanless, *Latin squares with no transversals*, Electron. J. Combin. 24 (2017), no. 2, #P2.45.
11. J. Dénes and A. D. Keedwell, *Latin squares and their applications*, Akadémiai Kiadó, Budapest, 1974.
12. A. A. Drisko, *Transversals in row-Latin rectangles*, J. Combin. Theory Ser. A. 84 (1998), 181–195.
13. P. Erdős, D. R. Hickerson, D. A. Norton, and S. K. Stein, *Unsolved problems: Has every Latin square of order $n$ a partial Latin transversal of size $n − 1$?* Amer. Math. Monthly. 95 (1988), 428–430.
14. A. B. Evans, *Maximal partial transversals in a class of Latin squares*, Australas. J. Combin. 73 (2019), 179–199.
15. A. B. Evans, A. Mammoliti, and I. M. Wanless, *Latin squares with maximal partial transversals of many lengths*, J. Combin. Theory Ser. A. 180 (2021), 105403.
16. P. Gao, R. Ramadurai, I. M. Wanless, and N. C. Wormald, *Full rainbow matchings in graphs and hypergraphs*, Combin. Probab. Comput. (2021), 1–19. https://doi.org/10.1017/S0963548320000620
17. L. Goddyn and K. Halasz, *All group-based Latin squares possess near transversals*, J. Combin. Des. 28 (2020), 358–365.
18. P. Hatami and P. W. Shor, *A lower bound for the length of a partial transversal in a Latin square*, J. Combin. Theory Ser. A. 115 (2008), 1103–1113.
19. P. Keevash, A. Pokrovskiy, B. Sudakov, and L. Yepremyan, *New bounds for Ryser’s conjecture and related problems*, arXiv:2005.00526, 2020.
20. B. D. McKay and I. M. Wanless, *On the number of Latin squares*, Ann. Comb. 9 (2005), 335–344.
21. R. Montgomery, A. Pokrovskiy, and B. Sudakov, *Decompositions into spanning rainbow structures*, Proc. Lond. Math. Soc. 119 (2019), no. 3, 899–959.
22. G. Perarnau and O. Serra, *Rainbow perfect matchings in complete bipartite graphs: existence and counting*, Combin. Probab. Comput. 22 (2013), 783–799.
23. A. Pokrovskiy and B. Sudakov, *A counterexample to Stein’s equi-n-square conjecture*, Proc. Amer. Math. Soc. 147 (2019), 2281–2287.
24. J. K. Pula, Approximate transversals of Latin squares, Ph.D. Thesis, University of Denver, 2011.
25. S. K. Stein, *Transversals of Latin squares and their generalizations*, Pacific J. Math. 59 (1975), 567–575.
26. S. K. Stein and S. Szabó, *The number of distinct symbols in sections of rectangular arrays*, Discrete Math. 306 (2006), 254–261.
27. I. M. Wanless, *Transversals in Latin squares: a survey*, Surveys in Combinatorics 2011, London Math. Soc. Lecture Notes 392 (R. Chapman), Cambridge University Press, Cambridge, (2011), pp. 403–437.
28. D. E. Woolbright, *An n × n Latin square has a transversal with at least n − √n distinct symbols*, J. Combin. Theory Ser. A 24 (1978), 235–237.

**How to cite this article:** D. Best, K. Pula, and I. M. Wanless, *Small Latin arrays have a near transversal*, J. Combin. Des. 2021;29:511–527. https://doi.org/10.1002/jcd.21778