Myrvold's Results on Orthogonal Triples of 10×10 Latin Squares: A SAT Investigation

Curtis Bright[1,3][0000−0002−0462−625X], Amadou Keita[2][0009−0001−5861−4617], and Brett Stevens[3][0000−0003−4336−1773]

[1] School of Computer Science, University of Windsor, Canada
[2] Department of Mathematics and Statistics, University of Windsor, Canada
[3] School of Mathematics and Statistics, Carleton University, Canada
{cbright,keitaa}@uwindsor.ca, brett@math.carleton.ca

Abstract. Ever since E. T. Parker constructed an orthogonal pair of 10×10 Latin squares in 1959, an orthogonal triple of 10×10 Latin squares has been one of the most sought-after combinatorial designs. Despite extensive work, the existence of such an orthogonal triple remains an open problem, though some negative results are known. In 1999, W. Myrvold derived some highly restrictive constraints in the special case in which one of the Latin squares in the triple contains a 4×4 Latin subsquare. In particular, Myrvold showed there were twenty-eight possible cases for an orthogonal pair in such a triple, twenty of which were removed from consideration. We implement a computational approach that quickly verifies all of Myrvold's nonexistence results and in the remaining eight cases finds explicit examples of orthogonal pairs—thus explaining for the first time why Myrvold's approach left eight cases unsolved. As a consequence, the eight remaining cases cannot be removed by a strategy of focusing on the existence of an orthogonal pair; the third square in the triple must necessarily be considered as well.

Our approach uses a Boolean satisfiability (SAT) solver to derive the nonexistence of twenty of the orthogonal pair types and find explicit examples of orthogonal pairs in the eight remaining cases. To reduce the existence problem into Boolean logic we use a duality between the concepts of *transversal representation* and *orthogonal pair* and we provide a formulation of this duality in terms of a composition operation on Latin squares. Using our SAT encoding, we find transversal representations (and equivalently orthogonal pairs) in the remaining eight cases in under a day of computing.

Keywords: Latin square · orthogonal Latin square · transversal representation · satisfiability solving.

1 Introduction

A *Latin square* of order n is an $n \times n$ array filled with n distinct symbols such that each symbol appears exactly once in each row and exactly once in each column. A *transversal* of a Latin square of order n consists of n cells of the square chosen so that there is exactly one cell from each row, exactly one cell from each column, and exactly n distinct symbols all together. There are many ways of representing a transversal, but we follow Myrvold [28] and represent a transversal by listing the symbols in the transversal in each column from left to right. For example, the highlighted transversal in $\begin{bmatrix} 0 & \mathbf{1} & 2 \\ \mathbf{2} & 0 & 1 \\ 1 & 2 & \mathbf{0} \end{bmatrix}$ is represented by the row vector $[2, 1, 0]$. We call this row vector the transversal's *row representation*.

Two Latin squares of order n are said to be *orthogonal* when all n^2 possible symbol pairs occur when the two squares are superimposed over each other. This is equivalent to each Latin square having a decompositon of its cells into transversals. A set of Latin squares that are pairwise orthogonal to each other are known as *mutually orthogonal Latin squares* (MOLS) and a set of k MOLS of order n are known as a k-MOLS(n). For each order n, let $N(n)$ denote the largest possible value of k for which a k-MOLS(n) exists. Determining values of $N(n)$ has a long history [1, Ch. III] and has been of intense interest to mathematicians ever since Euler conjectured in 1782 that $N(n) = 1$ for $n \equiv 2 \pmod 4$. It is easily seen that $N(2) = 1$, and Tarry showed in 1900 that $N(6) = 1$ [33]. However, in 1959, Euler's conjecture was shown to be false by the discovery of a 2-MOLS(22) [6] and a 2-MOLS(10) [29]. In fact, in 1960 it was shown that $N(n) \geq 2$ for all $n > 6$ [7]. It is also known that $N(n) = n - 1$ if and only if a projective plane of order n exists. Projective planes exist for all prime powers, so the first order for which the value of $N(n)$ is uncertain is $n = 10$. It is unknown if $N(10) \geq 3$, and determining the value of $N(10)$ is one of the most prominent unsolved problems concerning MOLS. In particular, finding a 3-MOLS(10) or proving its nonexistence is a longstanding open problem in combinatorial design theory.

Although it is not known if a 3-MOLS(10) exists or not, there are several special results known about this case. Mann [23] proved that a 10×10 Latin square with a 5×5 Latin subsquare cannot belong to an orthogonal pair, let alone an orthogonal triple. Parker [30] proved that two orthogonal 10×10 Latin squares with orthogonal 3×3 Latin subsquares cannot be part of an orthogonal triple. Myrvold [28] considered a 10×10 Latin square L with a 4×4 Latin subsquare. She showed that it *is* possible for L to be part of an orthogonal pair, and further considered if L can be part of an orthogonal triple. Myrvold showed there are seven possible ways of decomposing L into transversals and twenty-eight possibilities for decomposing L in two ways (a necessary condition for L to be part of an orthogonal triple). Myrvold ruled out the existence of twenty of the twenty-eight possibilities which required only the consideration of constraints arising from only two of the three putative squares. Her work left open the remaining eight cases:

> *The most obvious next step in extending the current work is to eliminate the remaining eight cases from consideration.* [28]

We provide a reason why Myrvold's method was unable to rule out these eight cases, and show any argument ruling out these cases must necessarily be more involved—because orthogonal pairs in the remaining eight cases exist (though it is unclear if orthogonal *triples* in the remaining eight cases exist). Thus, any argument ruling out the remaining eight cases must necessarily involve the triple as a whole, not only two of the three squares. We give more background on Latin squares and the formulation of Myrvold's twenty-eight cases in Section 2.

Our approach uses a satisfiability (SAT) solver to explicitly construct a 2-MOLS(10) in each of the eight cases that Myrvold left open. Additionally, in under a second of compute time the SAT solver shows the nonexistence of a 2-MOLS(10) in the twenty cases solved by Myrvold. To use a SAT solver, it is necessary to reduce the problem of searching for the object in question to the problem of searching for a satisfying assignment to a formula in Boolean logic representing Myrvold's framework and cases.

We reduce the problem of finding a 2-MOLS(10) in each of Myrvold's twenty-eight cases to SAT—see Section 4 for a description of our encoding. We develop a SAT encoding of orthogonality that relies on an equivalence between the orthogonality of Latin squares and what Myrvold calls a "transversal representation" Latin square [28]. Myrvold uses this equivalence for *"designing computer programs for exploring squares and their mates"*. We provide a precise duality relating these two concepts via a "composition" operation on Latin squares and a generalization of Latin squares where only the columns (and not necessarily the rows) contain all n symbols (see Section 3). This alternate "transversal representation" encoding allowed finding a 2-MOLS(10) for all of Myrvold's previously unsolved cases in a reasonable amount of computation. The hardest of the eight cases required at least 13 hours of compute time to solve—see Section 5 for more details.

2 Background

We define the notion of "transversal representation" and relate it to the orthogonality of Latin squares in Section 2.1, give a detailed explanation of the transversal representation types classified by Myrvold [28] in Section 2.2, and give a brief description of satisfiability solving in Section 2.3. Finally, we give a summary of related work in Section 2.4.

2.1 Transversals and Orthogonality

It is well-known that a Latin square has an orthogonal mate if and only if it can be decomposed into n disjoint transversals [35]. From the n disjoint transversals, an orthogonal mate can be formed by writing each transversal in its row representation and stacking the rows together. We call such a square a *transversal representation* of the orthogonal mate. An example of a 4×4 Latin square D with four disjoint transversals and the associated transversal representation D' is provided in Figure 1. The pair (D, D') is known as a *transversal representation pair.*

Although we are primarily interested in Latin squares, in the course of our investigations, we found that it was helpful to consider the more general case of "column-Latin" squares. A *column-Latin* square of order n is an $n \times n$ array filled with n distinct symbols and in which each column contains distinct symbols (and

$$D = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 0 & 3 \\ \hline 0 & 3 & 1 & 2 \\ \hline 2 & 1 & 3 & 0 \\ \hline 3 & 0 & 2 & 1 \\ \hline \end{array} \qquad D' = \begin{array}{|c|c|c|c|}\hline 0 & 1 & 2 & 3 \\ \hline 1 & 0 & 3 & 2 \\ \hline 2 & 3 & 0 & 1 \\ \hline 3 & 2 & 1 & 0 \\ \hline \end{array}$$

Fig. 1: A transversal representation pair of Latin squares of order four. Each transversal of D is highlighted in a different colour, and the row representations of the transversals are given in D'.

is thus a permutation), but the rows are not required to contain distinct symbols. *Row-Latin* squares are defined similarly: the rows of the square must contain distinct entries, but the columns might not [19]. It follows immediately that an $n \times n$ array filled with n distinct symbols is a Latin square if and only if it is both row-Latin and column-Latin. For our purposes, the usefulness of column-Latin squares stems from the fact that two column-Latin squares can be "composed" in a sensible way to form a third column-Latin square which preserves structure related to orthogonality (see Section 3). Thus, we state most of our results in terms of column-Latin squares.

The concept of orthogonality of Latin squares translates directly to column-Latin squares. However, the concept of transversal needs some modification. A "generalized transversal" of a column-Latin square of order n must still be a selection of n entries from each row and column, but the entries may not all be distinct. Figure 2 shows an example of this generalization; note the generalized transversals highlighted in D_1 contain duplicate entries and therefore are not traditional transversals. However, the row representation construction can still be used to construct the column-Latin square D_1' and we refer to the pair (D_1, D_1') as a transversal representation pair of column-Latin squares.

$$D_1 = \begin{array}{|c|c|c|c|}\hline 0 & 1 & 3 & 2 \\ \hline 1 & 3 & 2 & 0 \\ \hline 3 & 2 & 1 & 1 \\ \hline 2 & 0 & 0 & 3 \\ \hline \end{array} \qquad D_1' = \begin{array}{|c|c|c|c|}\hline 0 & 0 & 2 & 1 \\ \hline 1 & 1 & 1 & 3 \\ \hline 2 & 2 & 3 & 0 \\ \hline 3 & 3 & 0 & 2 \\ \hline \end{array}$$

Fig. 2: A transversal representation pair of 4×4 column-Latin squares. Note that the highlighted entries of D_1 are *not* transversals, but their row representations when placed in a 4×4 array do form a column-Latin square.

We now give purely logical definitions of "orthogonal pair" and "transversal representation" stated in a way that highlights the similarity between the concepts. Suppose $[a_0, \ldots, a_{n-1}]$ is a row representing a generalized transversal of a column-Latin square B. This means if i is a row index, j and j' are two distinct column indices, and $B[i,j] = a_j$, then $B[i,j'] \neq a_{j'}$ (otherwise, both the jth and j'th entries of the generalized transversal are in row i, which is not allowed in any transversal, generalized or not). Equivalently, if both $B[i,j] = a_j$ and $B[i,j'] = a_{j'}$, then the only possibility is that $j = j'$. This motivates the following definition.

Definition 1. *Let A and B be order n column-Latin squares. Row i of A* represents a transversal *of B when $A[i,j] = B[i',j]$ and $A[i,j'] = B[i',j']$ imply $j = j'$. The square A is said to be a* transversal representation *of B when each row of A represents a transversal of B, i.e., for all $0 \leq i, i', j, j' < n$,*

$$A[i,j] = B[i',j] \text{ and } A[i,j'] = B[i',j'] \text{ imply } j = j'.$$

Because Definition 1 is symmetric in A and B, A is a transversal representation of B if and only if B is a transversal representation of A. As before, we say (A,B) is a *transversal representation pair*.

On the other hand, if two column-Latin squares A and B are orthogonal this means that if (i,j) and (i',j') are two distinct (row, column) pairs then $(A[i,j], B[i,j]) \neq (A[i',j'], B[i',j'])$. Equivalently, it means that if both $A[i,j] = A[i',j']$ and $B[i,j] = B[i',j']$, the only possibility is that $(i,j) = (i',j')$. This motivates the following definition.

Definition 2. *Let A and B be order n column-Latin squares. A is said to be* orthogonal *to B if for all $0 \leq i, i', j, j' < n$,*

$$A[i,j] = A[i',j'] \text{ and } B[i,j] = B[i',j'] \text{ imply } j = j'.$$

Note that the equality of j and j' in Definition 2 also implies the equality of i and i' because A and B are column-Latin squares. The consequent in Definition 2 thus could equivalently been written as the more typical $(i,j) = (i',j')$, but we use the simpler $j = j'$ in order to highlight the striking similarity between Definitions 1 and 2.

2.2 Transversal Representation Types

We now review Myrvold's results [28] on the possible transversal representation types of a 10×10 Latin square L containing a 4×4 Latin subsquare Ω. Without loss of generality, we assume the subsquare appears in the bottom-right of L, i.e., in the rows and columns labeled 6 to 9. We also assume L consists of the symbols from the set $\{0, 1, 2, \dots, 9\}$ and Ω consists of symbols from the set $\{0, 1, 2, 3\}$. We partition the other regions of L into Δ (lower-left), Γ (upper-right), and Σ (upper-left) as shown in Figure 3. Since the subsquare Ω is a Latin square containing symbols from the set $\{0, 1, 2, 3\}$, the rectangles Δ and Γ must take symbols only from the set $\{4, 5, 6, \dots, 9\}$ and each row and column of Σ must contain exactly $6 - 4 = 2$ symbols from the set $\{4, 5, 6, \dots, 9\}$.

Suppose the cells with symbols in $\{0, 1, 2, 3\}$ are coloured white. A transversal of L can be of five possible forms depending on how many white cells it takes from the Latin subsquare Ω. A transversal containing i white cells from Ω (i.e., in

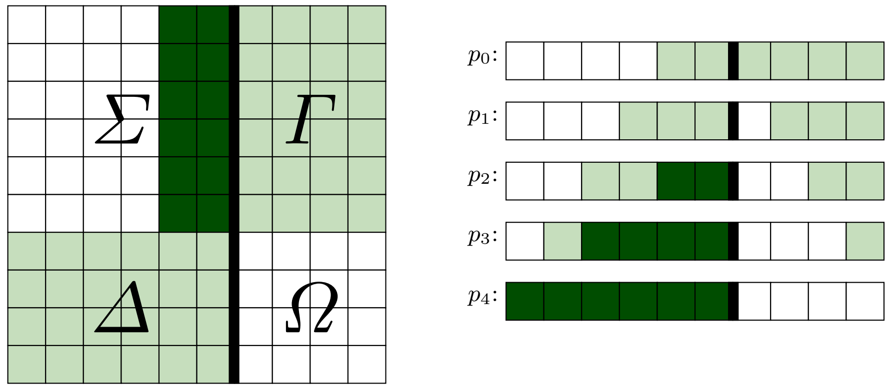

Fig. 3: The Latin square L (left) and its possible transversal types (right). White cells represent symbols in $\{0, 1, 2, 3\}$, light cells represent symbols in the rectangles Δ and Γ, and dark cells represent the symbols $\{4, 5, \ldots, 9\}$ in Σ. The cells of Σ are not shown in absolute positions; in actuality, each row and column of Σ has exactly two dark cells. Similarly, the transversal types are shown up to a permutation of the first six entries and the last four entries.

its last four columns) is said to be of form p_i (see Figure 3). Since any transversal will contain exactly four white cells in total, it must contain $4-i$ white cells in its first six columns. Consider the entries of p_i that were chosen from the first six rows of L (i.e., Σ or Γ). We have $4-i$ white entries (all from Σ) and $4-i$ entries from the last four columns of L (i.e., from Γ), so there are $6-2(4-i) = 2i-2$ remaining entries. The only possibilities for these are the nonwhite entries of Σ, and we colour these entries dark. This results in the following lemma.

Lemma 1 ([28, Lemma 3.1]). *A transversal of type p_i contains exactly $2i-2$ dark entries.*

A simple corollary of Lemma 1 is that p_0 is not a possible type, as it would have to contain -2 dark entries.

Let n_i be the number of transversals of type p_i in a transversal representation of L. Simple counting arguments give that the values $\{n_1, n_2, n_3, n_4\}$ satisfy the following Diophantine linear system.

$$\begin{aligned} n_i &\geq 0 && \text{nonnegativity of the counts,} \\ n_1 + n_2 + n_3 + n_4 &= 10 && \text{ten total transversals,} \\ n_1 + 2n_2 + 3n_3 + 4n_4 &= 16 && \text{sixteen total symbols in } \Omega. \end{aligned}$$

There are seven possible solutions to this linear system and correspondingly seven transversal representation types of L. These types are denoted R, S, T, U, V, W, and X by Myrvold. Table 1 gives the transversal type counts of each case.

Type	n_1	n_2	n_3	n_4
R	8	0	0	2
S	7	0	3	0
T	7	1	1	1
U	6	2	2	0
V	6	3	0	1
W	5	4	1	0
X	4	6	0	0

Table 1: A summary of the seven possible transversal types of L.

Up to ordering, there are $\binom{7}{2} = 21$ ways of choosing a pair with two different types, and 7 ways of choosing a pair with matching types, for a total of 28 possible transversal representation pair combinations. Under the assumption that L is part of an orthogonal triple, Myrvold [28, Thm 4.4] showed that the only possible pair types that can simultaneously be transversal representations of L are (X, S), (X, U), (X, V), (X, W), (X, X), (U, U), (U, W), and (W, W).

2.3 Satisfiability Solving

In this section, we provide some basic preliminaries on Boolean logic and satisfiability (SAT) solving. A *SAT solver* is a program that can determine if a Boolean logic formula can be *satisfied*—that is if there is a truth assignment under which the formula becomes true. In practice, the formulas provided to SAT solvers must be written in conjunctive normal form (CNF). Formulas in CNF only contain the Boolean connective operators $\land$ (and), $\lor$ (or), and $\neg$ (not). These operators have meanings similar to those in everyday English: the formula $x \land y$ is true if and only if both x and y are true; the formula $x \lor y$ is true if and only if x or y (or both) are true; and the formula $\neg x$ is true if and only if x is false.

A *literal* is a Boolean variable or its negation, i.e., a formula of the form x or $\neg x$ where x is a Boolean variable. A *clause* is a disjunction of literals, i.e., a formula of the form $l_1 \lor \cdots \lor l_k$ where $l_1, \ldots, l_k$ are literals. Finally, a formula is in *conjunctive normal form* when it is a conjunction of clauses, i.e., a formula of the form $c_1 \land \cdots \land c_k$ where $c_1, \ldots, c_k$ are clauses.

When A is a conjunction of literals and B is a disjunction of literals, we use the notation $A \rightarrow B$ as shorthand for $\neg A \lor B$. By basic logic equivalences, the formula $(\neg \bigwedge_i a_i) \lor \bigvee_i b_i$ is equivalent to $\bigvee_i \neg a_i \lor \bigvee_i b_i$, which (after applying the simplification $\neg\neg x \equiv x$ to any doubly negated literal) is a clause. Thus, we consider the notation $A \rightarrow B$ to be shorthand for a clause when B is a clause and A is a conjunction of literals.

Although there is no guarantee that SAT solvers can solve the SAT problem in a feasible amount of time, modern SAT solvers are highly effective at solving many kinds of SAT problems arising in practice [34], including mathematical problems such as the Boolean Pythagorean triples problem [14] and Lam's problem [9]. Many problems that at first seem unconnected to logic can be reduced to SAT problems due to the versatility of Boolean logic [10]. Consequently, SAT solvers are flexible tools that can be used for general-purpose search in many problems, including combinatorial ones.

2.4 Related Work

Extensive searches for a 3-MOLS(10) have been performed, and some important cases have been ruled out. For example, it is known that any such triple must only contain Latin squares with trivial symmetry groups [25]. Another computer-assisted proof showed that no orthogonal pairs of order ten have corresponding codes of dimension 33 [12]. Independent computer searches [9,18,31] have revealed that there is no projective plane of order ten, and because a projective plane of order n is equivalent to a $(n-1)$-MOLS(n) [8,27], these searches imply that no 9-MOLS(10)s exist or equivalently that $N(10) < 9$. Together with a result of Bruck [11], this implies that $N(10) \leq 6$ which is currently the best upper bound known on $N(10)$.

Egan and Wanless [13] enumerate MOLS of small orders, providing counts of orthogonal mates and classifications up to various equivalence notions for orders $n \leq 9$. They also present a set of three Latin squares L_1, L_2, L_3 of order 10 that is the closest known to forming a complete set of MOLS: L_1 is orthogonal to both L_2 and L_3, and 91 out of the 100 symbol pairs are different when L_2 and L_3 are superimposed. They also showed that L_2 and L_3 have seven common transversals.

Numerous studies have leveraged SAT solving, integer programming, and constraint programming in order to search for Latin squares of various forms such as diagonal Latin squares (a Latin square with with distinct symbols on its diagonal and anti-diagonal), and doubly self-orthogonal Latin squares (a Latin square orthogonal to its transpose and anti-transpose). Appa, Magos, and Mourtos [2,3] integrated integer programming and constraint programming to tackle the problem of searching for mutually orthogonal Latin squares. Their comparative study against traditional constraint and integer programming algorithms revealed the effectiveness of combining integer and constraint programming in searching for 2-MOLS(n) for $n \leq 12$ and 3-MOLS(n) for $n \leq 9$. Rubin et al. [32] formulated a symmetry breaking method and also provided an alternative constraint programming encoding based on a theorem of Mann [22] which performed much better in their search for pairs of orthogonal Latin squares. The SAT encoding that we use in our work can be viewed as a reformulation of their constraint programming encoding into Boolean satisfiability.

Ma and Zhang [21] use a general-purpose model searching program to find MOLS. They show a k-MOLS(n) exists if and only if there exists a Latin square of order n which has $k-1$ transversal matrices $T_1, \dots, T_{k-1}$ with any two transversal matrices T_i and T_j ($i \neq j$) being transversal matrices of each other [21, Prop 1]. As a result, instead of searching for k-MOLS(n), they searched for one Latin square and $k-1$ of its transversal matrices that are also transversal matrices of each other. They defined a Latin square L as a function $f \colon \mathcal{R} \times \mathcal{C} \to \mathcal{D}$ on row indices $\mathcal{R}$, column indices $\mathcal{C}$, and symbol set $\mathcal{D}$. Similarly, they defined the ith transversal matrix T_i $(1 \leq i \leq k-1)$ as a function $f_i \colon \mathcal{D}_i \times \mathcal{C} \to \mathcal{R}$, where $\mathcal{D}_i$ is the symbol set of L_i, the Latin square represented by the transversal matrix T_i. The formulae they used for encoding a k-MOLS(n) then consist of three types:

1. Formulae to specify that f and f_i are Latin squares:
$$f(x_1, y) = f(x_2, y) \to x_1 = x_2, \qquad f(x, y_1) = f(x, y_2) \to y_1 = y_2,$$
$$f_i(t_1, y) = f_i(t_2, y) \to t_1 = t_2, \qquad f_i(t, y_1) = f_i(t, y_2) \to y_1 = y_2.$$
2. Formulae to specify that f_i is a transversal matrix of f:
$$f(f_i(t, y_1), y_1) = f(f_i(t, y_2), y_2) \to y_1 = y_2.$$
3. Formulae to ensure that L_i and L_j are orthogonal by stating that T_i and T_j are a transversal representation pair:
$$\big(f_i(t_1, y_1) = f_j(t_2, y_1) \wedge f_i(t_1, y_2) = f_j(t_2, y_2)\big) \to y_1 = y_2.$$

Our encoding of "transversal representation pair" uses formulae that are similar to their first two types, though our encoding is purely represented as a Boolean satisfiability problem which doesn't natively support expressions like $f(f_i(t, y_1), y_1)$.

A Latin square that is orthogonal to both its transpose and its transpose across its anti-diagonal is known as a doubly self-orthogonal Latin square. For orders $n \equiv 2 \pmod 4$, the existence of such squares is unknown for $n > 10$. In 2011, Lu et al. [20] proved the nonexistence of a doubly self-orthogonal Latin square of order ten. They encoded the existence of a doubly self-orthogonal Latin square of order ten as a SAT problem and proved the nonexistence by showing the resulting SAT instance was unsatisfiable. To describe their encoding, let A be a self-orthogonal Latin square of order n, let A^T denote the transpose of A, and let A^* denote the transpose across the anti-diagonal of A, i.e., $A^T[x, y] = A[y, x]$ and $A^*[x, y] = A[n-1-y, n-1-x]$ where $0 \leq x, y < n$. In addition to the properties of a Latin square, they generated the constraints

$$\begin{aligned}
&(A[x_1, y_1] = A[x_2, y_2] \wedge A[y_1, x_1] = A[y_2, x_2]) \\
&\quad \to (x_1 = x_2 \wedge y_1 = y_2), \quad \text{i.e., orthogonality of } A \text{ and } A^T\text{, and} \\
&(A[x_1, y_1] = A[x_2, y_2] \wedge A[n-1-y_1, n-1-x_1] = A[n-1-y_2, n-1-x_2]) \\
&\quad \to (x_1 = x_2 \wedge y_1 = y_2), \quad \text{i.e., orthogonality of } A \text{ and } A^*.
\end{aligned}$$

A Costas array of order n is an $n \times n$ grid with n dots and $n^2 - n$ empty cells, with one dot in every row and column, and with no two dots sharing the same relative horizontal, vertical, or diagonal displacement. A Costas Latin square is a Latin square in which the cells for each symbol are a Costas array. Jin et al. [16] introduced an efficient method of using SAT solvers to search for Costas Latin squares. They established new existence and nonexistence results for various types of Costas Latin squares of even orders $n \leq 10$ including orthogonal pairs of Costas Latin squares. In their encoding, they define from square A a new square

TA by the rule $A[i,j] = k \to TA[k,j] = i$. This makes TA the $(3,2,1)$-*parastrophe* or the *column inverse* of A (cf. [17]), though they refer to TA as a transversal matrix. To encode orthogonality of (A, B), they impose the constraints

$$x \neq y \to (TA[u,x] \neq TB[v,x] \vee TA[u,y] \neq TB[v,y]) \quad \text{for } 0 \leq x, y, u, v < n.$$

In the rest of this paper, we will use the notation A^{-1} for the column inverse (see Section 3.1).

A Latin square of order n is idempotent when its diagonal consists of the entries 0, 1, …, n in order, and it is symmetric if it is equal to its own transpose. A golf design of order n is a collection of $n-2$ idempotent symmetric Latin squares of order n that are mutually disjoint, meaning that any two Latin squares in the collection share no common symbols in any cell (except for the cells along their diagonals). Two golf designs are *orthogonal* if every Latin square in one design has an orthogonal mate in the other design.

Huang et al. [15] investigated the existence of orthogonal golf designs via constraint programming and satisfiability testing. They reformulated the orthogonal mate finding problem as a transversal finding problem. They constructed the transversal matrix T of a Latin square L with the constraints

$$(y_1 = y_2 \vee L[T[x,y_1], y_1] \neq L[T[x,y_2], y_2]) \quad \text{for } 0 \leq x, y_1, y_2 < n,$$

and additionally used constraints specifying that T is a Latin square.

Diagonal Latin squares feature distinct symbols along both the main and back diagonals. Zaikin and Kochemazov [36] constructed SAT encodings to discover pairs of orthogonal diagonal Latin squares of order ten and pseudotriples of orthogonal diagonal Latin squares. A pseudotriple refers to a set of three Latin squares that nearly form an orthogonal triple, but the orthogonality condition is only required to hold on a subset of the cells of the Latin squares. They discovered a triple of diagonal Latin squares of order ten for which the orthogonality condition holds across 73 cells (the same 73 cells in each Latin square in the triple).

3 Composition and Duality

In this section, we describe a duality between the concepts of orthogonality and transversal representation. First, in Section 3.1 we define a "composition" operation on column-Latin squares. Then in Section 3.2 we use the composition operation to concisely characterize the duality.

3.1 Composition of Column-Latin Squares

A column-Latin square of order n can be denoted by $(f_0, f_1, \ldots, f_{n-1})$ where f_j represents the permutation of the jth column. For any two permutations f and g of the same length, the *composition* fg is another permutation where $(fg)(i) = f(g(i))$, i.e., applying g then f. The composition of two column-Latin squares $F = (f_0, \ldots, f_{n-1})$ and $G = (g_0, \ldots, g_{n-1})$ is defined as

$$FG = (f_0 g_0, \ldots, f_{n-1} g_{n-1}).$$

The (i, j)th entry of FG is then $f_j g_j(i) = F[G[i, j], j]$. The *column inverse* of a column-Latin square F, denoted F^{-1}, is the column-Latin square in which each column is the inverse permutation of the corresponding column of F.

Let e denote the identity column permutation with $e(i) = i$ for $0 \leq i < n$ and $E = (e, \ldots, e)$ the column-Latin square of order n formed by n copies of e. The following two lemmas appear in Laywine and Mullen [19, pp. 98–99], except stated in terms of row-Latin squares instead of column-Latin squares.

Lemma 2. *Let C be a column-Latin square. Then (C, E) is an orthogonal pair if and only if C is a Latin square.*

Lemma 3. *If $\{C_1, C_2, \ldots, C_m\}$ is a set of mutually orthogonal column-Latin squares, then for any column-Latin square G, the set $\{C_1G, C_2G, \ldots, C_mG\}$ comprises a set of mutually orthogonal column-Latin squares.*

The next proposition provides criteria that establish a necessary and sufficient condition for a Latin square to "witness" the orthogonality of two column-Latin squares. The biconditional statement in the proposition was proven by Mann [22] and also appears as Theorem 6.6 in [19], though we strengthen the proposition by showing that when the squares are Latin (not just column-Latin) the witness square arises as a transversal representation of one of the original two squares.

Proposition 1. *Let C and F be column-Latin squares. Then (C, F) is an orthogonal pair if and only if there is a Latin square Z such that $ZC = F$. Moreover, if in addition, C is a Latin square, then (Z, F) is a TRP.*

Proof. Suppose Z is a Latin square and $ZC = F$ for column-Latin squares C and F. By Lemma 2, (Z, E) is an orthogonal pair. By Lemma 3, (ZC, EC) is an orthogonal pair. Since $ZC = F$ and $EC = C$, it follows that (F, C) is an orthogonal pair.

Conversely, suppose (C, F) is an orthogonal pair. Let $Z = FC^{-1}$ (i.e., $ZC = F$). Since (C, F) is an orthogonal pair, by Lemma 3, (Z, E) is an orthogonal pair (since $FC^{-1} = Z$ and $CC^{-1} = E$). By Lemma 2, Z is a Latin square.

We now show that if C is a Latin square and F is a column-Latin square such that (C, F) is an orthogonal pair, then (Z, F), which is equal to (Z, ZC), is a TRP. Suppose that (Z, F) is not a TRP. Then there exist i, i', j, $j' \in \{0, 1, 2, \ldots, n-1\}$ where $j \neq j'$ with

$$Z[i, j] = ZC[i', j] = Z[C[i', j], j], \text{ and}$$
$$Z[i, j'] = ZC[i', j'] = Z[C[i', j'], j'].$$

Since Z is a Latin square, the symbols in each of its columns are distinct. Thus, considering the entries of column j of Z, we must have $C[i', j] = i$ and $C[i', j'] = i$, but $C[i', j] = C[i', j']$ is a contradiction because the rows of C (in particular, row i') are permutations, implying $j = j'$. Thus (Z, F) is a TRP. $\square$

3.2 Orthogonal Pair / Transversal Representation Duality

We now state a duality between orthogonality and transversal representations. This duality was already used by Myrvold [28, Thm 1.1], but we show how the duality can be concisely formulated in terms of the composition operation on column-Latin squares—a convenient viewpoint that we were unable to find in the literature. Roughly speaking, the following Lemmas 4 and 5 are the analogue of Lemmas 2 and 3 with "orthogonal pair" replaced by "transversal representation pair".

Lemma 4. *Let C be a column-Latin square. Then (C, E) is a TRP if and only if C is a Latin square.*

Proof. Let C be a column-Latin square and (C, E) be a TRP. It is enough to show that rows of C are each an n-permutation. Assume, for a contradiction, that this is not the case. Then for some $0 \leq i, j, j', k < n$ with $j \neq j'$, $C[i, j] = k = C[i, j']$. Since E is a transversal representation of C, row i of C has its tth symbol from column t of E. Therefore, the symbol k is on two different rows of E, which contradicts the definition of E. Therefore, rows of C are each an n-permutation, and consequently, C is a Latin square.

Conversely, suppose C is a Latin square. Since all symbols are distinct on each row of C and the same on each row of E, then each row of C takes symbols from distinct rows and columns of E and the tth symbol on each row is from column t of E. Thus E is a transversal representation of C. It follows that (C, E) is a TRP. □

Lemma 5. *Let $\{C_1, C_2, \ldots, C_m\}$ be a set of mutual TRPs of column-Latin squares, then for any column-Latin square G, the set $\{GC_1, GC_2, \ldots, GC_m\}$ comprises mutual TRPs.*

Proof. It is enough to prove this statement for a set of two column-Latin squares. The columns of GC_1 and GC_2 are compositions of two permutations, therefore GC_1 and GC_2 are column-Latin squares. Assume, for a contradiction, that this is not the case. Suppose there exist i, i', j, $j' \in \{0, 1, 2, \ldots, n-1\}$ where $j \neq j'$ with

$$GC_1[i, j] = GC_2[i', j] \text{ and } GC_1[i, j'] = GC_2[i', j'].$$

Thus by equality of the symbols

$$G[C_1[i, j], j] = G[C_2[i', j], j] \text{ and } G[C_1[i, j'], j'] = G[C_2[i', j'], j'].$$

Since G is a column-Latin square, the uniqueness of symbols in its columns provides that

$$C_1[i, j] = C_2[i', j] \text{ and } C_1[i, j'] = C_2[i', j'].$$

Since (C_1, C_2) is a TRP, we have $j = j'$. This contradicts our assumption. Thus (GC_1, GC_2) is a TRP. Therefore, the set consists of mutual TRPs. □

Proposition 2. *Let C and F be column-Latin squares. Then (C, F) is a TRP if and only if there is a Latin square Z such that $CZ = F$. Moreover, if C is a Latin square, then Z is orthogonal to F.*

Proof. Assume there exists a Latin square Z such that $CZ = F$. By Lemma 4, (Z, E) is a TRP. By Lemma 5, (C, F), which is equal to (CE, CZ), is a TRP.

Conversely, assume (C, F) is a TRP. Let $Z = C^{-1}F$. Since (C, F) is a TRP and $(C^{-1}C, C^{-1}F) = (E, Z)$, by Lemma 5, (E, Z) is a TRP. Thus (E, Z) is a TRP. We have that Z is a Latin square by Lemma 4.

Now we prove that if C is a Latin square, Z and F are orthogonal. Assume, for a contradiction, that (Z, F) (where $F = CZ$) is not an orthogonal pair, i.e., there exist i, i', j, $j' \in \{0, 1, 2, \ldots, n-1\}$ with $j \neq j'$ for which

$$Z[i, j] = Z[i', j'] \text{ and } F[i, j] = F[i', j'].$$

The second equation implies $C[Z[i, j], j] = C[Z[i', j'], j']$ an equality between two symbols in rows j and j' of C, which, after using the first equation, yields $C[Z[i, j], j] = C[Z[i, j], j']$. Since C is a Latin square, its rows are permutations, which implies $j = j'$ and contradicts the assumption that $j \neq j'$. Therefore, (Z, F) must be an orthogonal pair. □

The following result describes the equivalence between a set of mutually orthogonal column-Latin squares and a set of mutually TRPs. The correctness of our SAT encoding relies on this equivalence.

Theorem 1 (cf. [28]). *Let $\mathcal{C}$ denote the set $\{C_1, \ldots, C_r\}$ of r column-Latin squares of order n.*

(a) If $\mathcal{C}$ contains mutually orthogonal squares, then the set

$$\{\, Z_1, \ldots, Z_r : Z_1 = C_1, Z_t = C_1 C_t^{-1} \text{ for } 2 \leq t \leq r \,\}$$

contains mutual TRPs.

(b) If $\mathcal{C}$ consists of mutual TRPs, then the set

$$\{\, Y_1, \ldots, Y_r : Y_1 = C_1, Y_t = C_t^{-1} C_1 \text{ for } 2 \leq t \leq r \,\}$$

contains mutually orthogonal pairs.

Proof. For (a), suppose the set $\{\, C_i : 1 \leq i \leq r \,\}$ consists of mutually orthogonal column-Latin squares of order n. Construct a set of r squares $\{\, Z_i : 1 \leq i \leq r \,\}$ by letting $Z_1 = C_1$ and $Z_t = C_1 C_t^{-1}$ for $2 \leq t \leq r$. Proposition 1 gives that each Z_t, $2 \leq t \leq r$ is a Latin square; further it ensures that (Z_1, Z_t) is a TRP. Observe that $Z_t C_t C_s^{-1} = Z_s$ for $2 \leq t, s \leq r$ where $t \neq s$. Since both C_t and C_s^{-1} are column-Latin squares, their composition is a column-Latin square. Thus (Z_t, Z_s) for $2 \leq t, s \leq r$ where $t \neq s$, being a TRP also follows from Proposition 1.

For (b), suppose the set $\{\, C_i : 1 \leq i \leq r \,\}$ consists of column-Latin squares of order n such that any two squares form a TRP. Construct a set of r squares $\{\, Y_i : 1 \leq i \leq r \,\}$ by letting $Y_1 = C_1$ and $Y_t = C_t^{-1} C_1$ for $2 \leq t \leq r$. Proposition 2 gives that each Y_t, $2 \leq t \leq r$ is a Latin square; and that Y_1 and Y_t are orthogonal. Observe that $C_s^{-1} C_t Y_t = Y_s$ for $2 \leq t, s \leq r$ where $t \neq s$. Since both C_s^{-1} and C_t are column-Latin squares, their composition is a column-Latin square. Therefore, Y_t being orthogonal to Y_s for $2 \leq t, s \leq r$ where $t \neq s$ also follows from Proposition 2. □

4 Encoding and Implementation

In this section we describe our encoding of the problem of constructing transversal representation pairs (TRPs) into a Boolean satisfiability problem and how we use our encoding to search for TRPs for each of Myrvold's 28 possible types described in Section 2.2. Recall that Myrvold's 28 types describe TRPs (P, Q) for which P and Q are each transversal representations of a Latin square L of order $n = 10$ containing a 4×4 Latin subsquare.

To reduce the existence of the $n \times n$ square P into Boolean logic, we use n^3 Boolean variables $P_{i,j,k}$ (for $0 \leq i, j, k < n$) with $P_{i,j,k}$ denoting the fact that the (i, j)th entry of P is k. Similarly, another n^3 Boolean variables $Q_{i,j,k}$ for $0 \leq i, j, k < n$ represent the entries of the square Q.

Once these variables have been defined, we need to specify constraints that P and Q are Latin squares (see Section 4.1), are a transversal representation pair (see Section 4.2), and conform to one of Myrvold's 28 types (see Section 4.3). We also describe a method of symmetry breaking which reduces the size of the search space by adding additional constraints which hold without loss of generality (see Section 4.4). Finally, once we have found a collection of TRPs, we run a postprocessing step on them, ensuring that the TRPs are pairwise inequivalent and that they cannot be extended to a set of three mutual TRPs (see Section 4.5).

Our encoding scripts were written in Python and are freely available at `https://github.com/curtisbright/Myrvold-MOLS`.

4.1 Latin Square Constraints

First, we need to describe constraints on the variables $P_{i,j,k}$ (meaning that $P[i, j] = k$) which assert that P is a Latin square. Direct methods for doing this from the definition of a Latin square are well known and widely used; e.g., see (10.1)–(10.4) in Zhang's survey [37]. The direct method asserts that every cell of P contains *at least one* symbol and *at most one* symbol, i.e.,

$$\bigvee_{0 \leq i < n} P_{p,q,i} \quad \text{and} \quad \bigwedge_{0 \leq i < j < n} (\neg P_{p,q,i} \vee \neg P_{p,q,j}) \quad \text{for all } 0 \leq p, q < n.$$

Additionally, every column of P contains n distinct symbols,

$$\bigvee_{0 \leq i < n} P_{i,q,r} \quad \text{and} \quad \bigwedge_{0 \leq i < j < n} (\neg P_{i,q,r} \vee \neg P_{j,q,r}) \quad \text{for all } 0 \leq q, r < n,$$

and similarly every row of P contains n distinct symbols,

$$\bigvee_{0\leq i<n} P_{p,i,r} \quad \text{and} \quad \bigwedge_{0\leq i<j<n} (\neg P_{p,i,r} \vee \neg P_{p,j,r}) \quad \text{for all } 0 \leq p, r < n.$$

Such an encoding uses $3n + 3\binom{n}{2}$ clauses and is known as the "binomial" or "pairwise" encoding of the *exactly one* predicate [24]. While this encoding gave good performance, in our experiments we got slightly better performance with the *cardinality* encoding of Bailleux and Boufkhad [4]. Their encoding reduces a constraint like $x_1 + \cdots + x_n = r$ (where r is a fixed integer between 0 and n and we think of the Boolean x_is as $\{0,1\}$ variables) into conjunctive normal form. Using this encoding we specify that P is a Latin square with the cardinality constraints

$$\sum_{0\leq i<n} P_{p,q,i} = 1, \quad \sum_{0\leq i<n} P_{i,p,q} = 1, \quad \sum_{0\leq i<n} P_{p,i,q} = 1 \quad \text{for all } 0 \leq p, q < n,$$

and a similar encoding can be used to specify that Q is also a Latin square.

4.2 Transversal Representation Constraints

The direct encoding that (P, Q) is a TRP using the contrapositive of Definition 1 would be

$$(P_{i,j,k} \wedge P_{i,j',l} \wedge Q_{i',j,k}) \to \neg Q_{i',j',l} \quad \text{for all } 0 \leq i, i', j, j', k, l < n \text{ with } j < j'.$$

This is because if row i of P has its jth entry as k and its (j')th entry as l, then in whatever row of Q which has its jth entry as k (one such row must exist since Q is a Latin square) that row *cannot* have its (j')th entry as l, or that row wouldn't represent a transversal. However, this encoding uses $n^4\binom{n}{2} = \Theta(n^6)$ clauses of length 4 which is not ideal in practice. Instead, our encoding that (P, Q) is a TRP will assert the existence of the Latin square $Z = P^{-1}Q$ and by Proposition 2 this implies that P and Q are a transversal representation pair.

As before, the entries of the square Z are encoded via n^3 new variables $Z_{i,j,k}$ (with $0 \leq i, j, k < n$) and Z is enforced to be a Latin square using the same encoding described in Section 4.1. Now we need to enforce the relationship $Q = PZ$, which means that the (i, j)th entry of Q is equal to the (i', j)th entry of P, where $i' = Z[i, j]$. Letting k represent the (i, j)th entry of Q, this gives the constraints

$$(Z_{i,j,i'} \wedge P_{i',j,k}) \to Q_{i,j,k} \quad \text{for all } 0 \leq i, i', j, k < n.$$

Moreover, because $P = QZ^{-1}$ and $Z = QP^{-1}$, we similarly derive the constraints

$$(Z_{i,j,i'} \wedge Q_{i,j,k}) \to P_{i',j,k} \quad \text{for all } 0 \leq i, i', j, k < n,$$
$$(P_{i',j,k} \wedge Q_{i,j,k}) \to Z_{i,j,i'} \quad \text{for all } 0 \leq i, i', j, k < n.$$

These last two kinds of constraints are technically redundant, but we found that they tended to improve the performance of the solving in practice.

Thus, our encoding that (P, Q) is a TRP uses $3n^4$ clauses and the $3n^2$ cardinality constraints $\sum_i Z_{i,j,k} = \sum_i Z_{j,k,i} = \sum_i Z_{j,i,k} = 1$ for all $0 \leq j, k < n$. Altogether, this TRP encoding uses $\Theta(n^4)$ clauses of length at most 3, and in practice this is preferable to the $\Theta(n^6)$ clauses of length 4 used by the direct encoding.

A similar $\Theta(n^4)$ clause encoding was previously derived by Zhang (see [37, Lemma 2]), for ensuring the orthogonality of a pair (A, B) of Latin squares of order n. Zhang's encoding for orthogonality uses a new predicate $\Phi(i, j, k)$ introduced via a clever trick and Zhang mentions that *"It is a challenge to develop a method which can automatically generate the predicates like Φ..."* [38]. Zhang essentially uses constraints saying that the "columns" of Φ have distinct symbols and that the entries of A and B determine the "entries" of Φ. Following our notation, Zhang uses constraints of the form

$$(A_{ijk} \wedge B_{ijl}) \rightarrow \Phi(i, k, l), \qquad \text{for all } 0 \leq i, j, k, l < n.$$

In light of the above and Proposition 1, this means that not only is Φ itself a Latin square, it can be naturally viewed as a transversal representation of one of the original Latin squares and conveniently expressed via a composition square.* Viewing Φ as a composition square, one can derive additional constraints using this extra structure on Φ (e.g., the entries of A and Φ determine the entries of B). While such constraints are technically redundant, they tended to help the efficiency of the solver, at least in the experiments that we performed.

4.3 Colour Constraints

We now describe how we encode that the square P is one of Myrvold's eight types as described in Table 1 (and an identical encoding can be used for Q). In order to do this, we need to be able to specify the *colour* of each cell in the square P to be either white, light, or dark. Let $\mathtt{w}$ and $\mathtt{d}$ represent fixed symbols which are not in our symbol set $\{0, \ldots, n-1\}$.

We let the Boolean variable $P_{i,j,\mathtt{w}}$ represent the (i, j)th entry of P is white, and let the Boolean variable $P_{i,j,\mathtt{d}}$ represent that the (i, j)th entry of P is dark. Otherwise, if both $P_{i,j,\mathtt{w}}$ and $P_{i,j,\mathtt{d}}$ are false, then the (i, j)th entry of P will be light. Note that dark variables are only necessary in the first six columns, since no dark entries appear in the last four columns (see Figure 3). Additionally, the position of darks in the first six columns completely determine the position of whites in the first six columns—the whites will be the cells with symbols $\{4, \ldots, 9\}$ not coloured black—making the variables $P_{i,j,\mathtt{w}}$ only necessary for $j \geq 6$. Altogether, we introduce n^2 new variables encoding the colours of P.

*In fact, the specific constraints used by Zhang causes the *columns* of Φ to represent transversals of B and for Φ to be the composition square BA^{-1} where the composition and inverse is defined *row-wise* instead of column-wise like in the rest of this paper.

To ensure the symbols $\{0, \ldots, 3\}$ are coloured white, we use the clauses

$$P_{i,j,r} \rightarrow P_{i,j,\mathtt{w}} \quad \text{for all } 0 \leq i < n,\ 6 \leq j < n, \text{ and } 0 \leq r < 4,$$

and conversely to ensure that only symbols $\{0, \ldots, 3\}$ are coloured white we use $P_{i,j,\mathtt{w}} \rightarrow \bigvee_{0 \leq r < 4} P_{i,j,r}$ for all $0 \leq i < n$ and $6 \leq j < n$. Similarly, to ensure that only symbols $\{4, \ldots, 9\}$ are coloured dark, we use the clauses

$$P_{i,j,\mathtt{d}} \rightarrow \bigvee_{4 \leq r < n} P_{i,j,r} \quad \text{for all } 0 \leq i < n \text{ and } 0 \leq j < 6.$$

Recall that a transversal is said to be of type p_k when it has k whites in its last four entries. By Lemma 1, p_k will also have $2k - 2$ dark entries in its first six entries. Thus, in order to specify that row i in P is of type p_k, we use the constraints

$$\sum_{0 \leq j < 6} P_{i,j,\mathtt{d}} = 2k - 2 \quad \text{and} \quad \sum_{6 \leq j < n} P_{i,j,\mathtt{w}} = k.$$

Here, like in Section 4.1, we think of Boolean variables as taking $\{0, 1\}$ values and encode the cardinality constraints with the encoding of Bailleux and Boufkhad [4]. We also know that each of the first six columns of P contain exactly two dark entries, so we use the cardinality constraints

$$\sum_{0 \leq i < n} P_{i,j,\mathtt{d}} = 2 \quad \text{for all } 0 \leq j < 6.$$

Similarly, we also use n^2 Boolean variables $Q_{i,j,\mathtt{w}}$ and $Q_{i,j,\mathtt{d}}$ to represent the colours of the square Q and add similar constraints to those above (using the $Q_{i,j,\mathtt{w}}$ and $Q_{i,j,\mathtt{d}}$ variables in place of the $P_{i,j,\mathtt{w}}$ and $P_{i,j,\mathtt{d}}$ variables). We now have specified a coloured TRP (P, Q) with each of P and Q conforming to any of Myrvold's types R, S, …, X that we select in advance. However, because P and Q are both transversal representations of the same coloured square L, it is important that their colours are *consistent* between themselves. In particular, the two entries coloured dark in each of the first six columns of P must match the two entries coloured dark in each of the first six columns of Q. (The white colours always match as they correspond exactly to the symbols $\{0, 1, 2, 3\}$, so if the black colours match then so must the light colours.)

Suppose the (i, j)th entry of P had symbol k and was coloured dark. Then, in order for the colouring to be consistent, the entry of Q in the jth column having symbol k must also be coloured dark. Note that symbol k must exist in the jth column of Q because Q is a Latin square, so say this happens in row i'. Then to express the consistency of the colours in P and Q we use the constraints

$$(P_{i,j,k} \wedge P_{i,j,\mathtt{d}} \wedge Q_{i',j,k}) \rightarrow Q_{i',j,\mathtt{d}} \quad \text{for all } 0 \leq i, i' < n,\ 0 \leq j < 6, \text{ and } 4 \leq k < n.$$

Although not strictly necessary, we also add the same constraints except deriving colour of cell (i, j) in P from the colour of cell (i', j) in Q. This gives the constraints

$$(P_{i,j,k} \wedge Q_{i',j,\mathtt{d}} \wedge Q_{i',j,k}) \rightarrow P_{i,j,\mathtt{d}} \quad \text{for all } 0 \leq i, i' < n,\ 0 \leq j < 6, \text{ and } 4 \leq k < n.$$

4.4 Symmetry Breaking

The ordering of rows of a transversal representation square is arbitrary in the sense that if P is a transversal representation of Q, then the rows of P can be freely permuted while preserving the fact that it is a transversal representation of Q. Similarly, the rows of Q may also be permuted. Columns may not be permuted independently, but if (P, Q) is a TRP and the same permutation of columns is applied to both P and Q simultaneously, then the resulting new pair will also be a TRP. Similarly, the same permutation of symbols applied to both squares in a TRP maintains the property of the pair being a TRP. By a *coloured* TRP we mean one whose cells have been assigned the colours $\{\text{white}, \text{light}, \text{dark}\}$ corresponding to Myrvold's types described in Section 2.2. Permuting the rows or columns of a coloured TRP will permute its colours, but permutation of symbols will not permute colours. However, because symbols $\{0, 1, 2, 3\}$ are always coloured white in this paper, we only consider symbol permutations fixing the symbols $\{0, 1, 2, 3\}$ amongst themselves.

We say that two TRPs (P, Q) are *equivalent* when one can be generated from the other by applying symbol permutations to both squares simultaneously, column permutations to both squares simultaneously, or row permutations to either square independently. We will restrict ourselves to these equivalence operations, though other more involved equivalence operations on TRPs are possible.* The above equivalence operations generate a group of size $(n!)^4$, meaning that the search space contains a large number of symmetries which artificially increase its size. Ideally, we would like to add constraints to the search space in order to limit the search to just one representative from each equivalence class—this is known as a *perfect symmetry break*. Although we do not achieve the goal of a perfect symmetry break, we are able to remove most symmetries from the search by only searching for TRPs in what we call *standard type*.

Definition 3. *A coloured TRP (P, Q) is in* standard type *if the rows of each square are sorted by transversal type (i.e., if row i has type p_k and row $i' \geq i$ has type $p_{k'}$ then $k \leq k'$), the final row of P is one of*

q_2:	2	3	4	5	6	7	0	1	8	9
q_3:	3	4	5	6	7	8	0	1	2	9
q_4:	4	5	6	7	8	9	0	1	2	3

and all rows of P (except the last) and all rows of Q are sorted in increasing lexicographic order when they have the same transversal type.

In the following result, we demonstrate that every equivalence class of TRPs contains at least one TRP in standard type.

*For example, if (P, Q) is a TRP then (Z, Q) (with $Z = P^{-1}Q$) is an orthogonal pair by Proposition 2, so (Z^T, Q^T) is also an orthogonal pair, and so $(Z^T(Q^T)^{-1}, Q^T)$ is a TRP by Proposition 1.

Theorem 2. *Every coloured TRP is isomorphic to a coloured TRP in standard type.*

Proof. Let (Z, Z') be an arbitrary coloured TRP that we want to transform to a pair (Y, Y') in standard type. First, permute the rows of Z to put together rows of the same transversal type p_i (for $i \in \{1, 2, 3, 4\}$) such that all rows of type p_k come before all rows of type $p_{k'}$ when $k < k'$. Next, permute the rows of Z' in a similar fashion so the rows of Z' are also sorted by transversal type.

Suppose the last row of Z (i.e., Z_{n-1}), is of transversal type p_m for $m \in \{2, 3, 4\}$ (note that $m = 1$ is not possible due to the previous sorting of the rows of Z). If $m = 2$, permute the columns of Z and Z' simultaneously to colour light columns two, three, eight, and nine of Z_{n-1}, and to colour dark columns four and five of Z_{n-1}. If $m = 3$, permute the columns of Z and Z' simultaneously to colour light columns one and nine of Z_{n-1}, and colour dark columns two to five of Z_{n-1}. If $m = 4$, permute the columns of Z and Z' simultaneously to colour dark columns zero to five of Z_{n-1}.

Afterward, apply symbol permutations to Z and Z' simultaneously to fix a symbol assignment for Z_{n-1}. If $m = 2$, fix the last row as $\boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{6}\boxed{7}\boxed{0}\boxed{1}\boxed{8}\boxed{9}$; if $m = 3$, fix the last row as $\boxed{3}\boxed{4}\boxed{5}\boxed{6}\boxed{7}\boxed{8}\boxed{0}\boxed{1}\boxed{2}\boxed{9}$; and if $m = 4$, fix the last row as $\boxed{4}\boxed{5}\boxed{6}\boxed{7}\boxed{8}\boxed{9}\boxed{0}\boxed{1}\boxed{2}\boxed{3}$.

Finally, within each subset of rows of the same transversal type in the first $n - 1$ rows of Z and all the n rows of Z', permute the rows so they appear in increasing lexicographic order. Note that the final row Z_{n-1} cannot be so permuted without disturbing its entries which have already been fixed. □

Thus, without loss of generality we can assume the TRP we are searching for is in standard type and so we add extra constraints into our encoding to enforce this. Fixing the last row of P is easy by adding appropriate unit clauses (i.e., clauses of length 1), namely,

$$\text{for } q_2, \quad \bigwedge_{j=0}^{5} P_{9,j,j+2} \wedge \bigwedge_{j=6}^{7} P_{9,j,j-6} \wedge \bigwedge_{j=8}^{9} P_{9,j,j} \wedge \bigwedge_{j=4}^{5} A_{9,j,\mathtt{d}};$$

$$\text{for } q_3, \quad \bigwedge_{j=0}^{5} P_{9,j,j+3} \wedge \bigwedge_{j=6}^{8} P_{9,j,j-6} \wedge P_{9,9,9} \wedge \bigwedge_{j=2}^{5} A_{9,j,\mathtt{d}};$$

$$\text{for } q_4, \quad \bigwedge_{j=0}^{5} P_{9,j,j+4} \wedge \bigwedge_{j=6}^{9} P_{9,j,j-6} \wedge \bigwedge_{j=0}^{5} A_{9,j,\mathtt{d}}.$$

Enforcing the fact that rows are sorted by transversal type is done with the cardinality constraints discussed in Section 4.3, as these constraints allow us to fix which rows are of which types. For example, suppose that P is of type R, meaning that P consists of eight transversals of type p_1 and two transversals of type p_4. Then we would enforce the first eight rows of P to be of type p_1 with $P_{i,6,\mathtt{w}} + \cdots + P_{i,9,\mathtt{w}} = 1$ and $P_{i,0,\mathtt{d}} + \cdots + P_{i,5,\mathtt{d}} = 0$ for $0 \leq i < 8$, and the last two rows of P to be of type p_4 with $P_{i,6,\mathtt{w}} + \cdots + P_{i,9,\mathtt{w}} = 4$ and $P_{i,0,\mathtt{d}} + \cdots + P_{i,5,\mathtt{d}} = 6$ for $i = 8$ and 9.

Finally, we enforce that rows with the same transversal type in Q are sorted in lexicographic order by ensuring their initial entries are increasing. For example, suppose rows i and $i+1$ of Q have the same transversal type. Then we add the constraint $Q_{i,0,k} \to \neg Q_{i+1,0,l}$ for all $0 \leq l < k < n$, which says that the initial entry of row $i+1$ cannot be smaller than the initial entry of row i. We add the same constraints for P as well, except we only add the constraints for rows i and $i+1$ of the same transversal type with $i < 8$ in order to not add a lexicographic constraint on the final row of P.

4.5 Postprocessing

As we will describe in Section 5, the encoding presented thus far successfully found many TRPs (P, Q) corresponding to Myrvold's eight unsolved cases. We performed some postprocessing on these pairs to check if they were extendable to a triple of mutual transversal representations and also to check the pairs for equivalence.

First, we used a SAT solver to check all pairs (P, Q) for extendability to a triple. This was done by creating new SAT instances for each pair encoding both squares P and Q, along with a new Latin square L, and then asserting that (L, P) is a TRP and (L, Q) is a TRP by using the encoding described in Section 4.2 twice. The entries of P and Q were specified using unit clauses; i.e., if $P[i, j] = k$ then the clause $P_{i,j,k}$ was added to the SAT instance. Because of the presence of so many unit clauses these instances were highly constrained and in all cases were shown by the SAT solver to be unsatisfiable within 0.1 seconds. Thus, no pairs we found were extendable to a triple. However, this does not eliminate the possibility that there might exist a triple (P, Q, L) corresponding to some of Myrvold's cases, because we did not exhaustively enumerate all (P, Q)s for any of Myrvold's unsolved types.

Finally, we checked all the TRPs (P, Q) that we found to see if any were equivalent to each other. This was done by converting the TRP into its orthogonal pair representation $(P^{-1}Q, Q)$, reducing the orthogonal pair to a graph using the reduction given by Egan and Wanless [13], and finally checking the graphs for equivalence using the graph isomorphism tool *nauty* [26].

Precisely, the reduction from a $(k-2)$-MOLS(n) to a graph is described using what is known as an orthogonal array. An orthogonal array for a $(k-2)$-MOLS(n) is a matrix O of size $n^2 \times k$, with entries in $\{0, \dots, n-1\}$, with every possible pair of symbols appears exactly once in any two columns of O. Define an undirected graph G_O corresponding to O. The vertices of G_O are of three types:

- k type 1 vertices that correspond to the columns of O,
- kn type 2 vertices that correspond to the symbols in each of the columns of O, and
- n^2 type 3 vertices that correspond to the rows of O.

	# solved	mean	median	minimum	maximum
(U, U)	45/45	35305.9	17664.5	1711.1	168816.2
(U, W)	45/45	78597.0	46405.1	1916.1	230692.4
(V, X)	45/45	81822.4	42871.7	2532.5	462846.9
(S, X)	45/45	102768.9	56155.6	1882.8	449225.4
(U, X)	45/45	104215.3	78370.3	289.5	494242.1
(W, W)	42/45	248711.1	139056.9	6913.3	timeout
(W, X)	38/45	387779.8	252130.0	8735.5	timeout
(X, X)	13/45	951393.5	timeout	48016.3	timeout

Table 2: A summary of the running times (in seconds) of forty-five SAT instances for each of the eight pair types with solutions. The timeout was one week.

Each type 1 vertex is joined to the n type 2 vertices that correspond to the symbols in its column. Each type 3 vertex is connected to the k type 2 vertices that correspond to the symbols in its row. Vertices are coloured according to their type so that isomorphisms are not allowed to change the type of a vertex.

After forming the graphs corresponding to all TRPs (P, Q) we found, nauty determined that no two graphs were isomorphic. Thus, we have confirmation that the SAT solver was indeed exploring different parts of the search space and that multiple inequivalent TRPs exist corresponding to Myrvold's unsolved cases. However, we did not attempt to perform an exhaustive search for TRPs in any of Myrvold's unsolved cases. Given the enormity of the search space, and the fact that no solutions were repeated even after several hundred solutions had already been found, makes us suspect that an exhaustive search would require a huge amount of additional computational resources or at least some more restrictive properties that could be applied to Myrvold's unsolved cases.

5 Results

We now discuss the results of our computational investigation into Myrvold's results. The computations were performed using the SAT solver Kissat 4.0 [5] run on Intel Gold 6148 Skylake processors running at 2.4 GHz and equipped with 4 GiB of memory.

Recall Myrvold showed [28, Thm 4.4], if P and Q are both transversal representations of a Latin square of order ten containing a subsquare of order four, then up to ordering there are twenty-eight possible cases for P and Q and twenty of these cases can be ruled out. The eight possible cases Myrvold left remaining are (S, X), (U, X), (V, X), (W, X), (X, X), (U, U), (U, W), and (W, W).

We used our SAT encoding to generate twenty-eight SAT instances, one for each of Myrvold's cases. The twenty cases ruled out by Myrvold were each found to be unsatisfiable in under 0.2 seconds. The eight cases left open by Myrvold were all considerably harder to solve but each were found to be satisfiable, explaining why Myrvold was unable to eliminate these eight cases from consideration. Kissat stops solving as soon as it finds a satisfying assignment of the provided instance, and we use the satisfying assignment reported by Kissat to form a coloured TRP in each of the eight cases (see the Appendix for explicit examples of TRPs in each case).

Because the satisfiable cases were significantly more difficult than the unsatisfiable cases, we found it useful to exploit parallelization when solving the satisfiable instances. We started fourty-five independent Kissat processes for each satisfiable case and each process was run on one core of a Skylake CPU for up to one week. Each process was provided with a different random seed, so no two copies of Kissat would make the same choices during the solving process. Each process was terminated if Kissat did not find a solution within a week. Results from these searches are available in Table 2 and a scatterplot of the running times are plotted in Figure 4.

Although there is a significant amount of variance in the running times it was evident that some instances were more difficult to solve than others. In general, the case (U,U) was the easiest to solve and the case (X,X) was the hardest to solve.

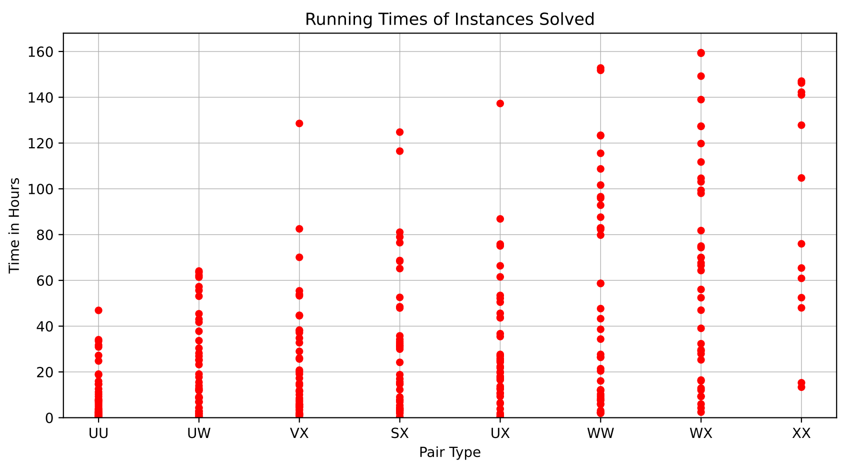

Fig. 4: A scatter plot of the runtimes.

6 Conclusion

In this paper we use a satisfiability (SAT) solver to investigate Myrvold's nonexistence results [28] on orthogonal triples of Latin squares of order ten. The SAT solver automatically rules out the cases that Myrvold ruled out, and more significantly, the SAT solver provides explicit examples of Latin square pairs in each of the cases that Myrvold was unable to rule out—providing an explanation for why

Myrvold was unable to rule out these cases and determining a negative resolution to the following question left open by Myrvold:

Possibly, with a bit more ingenuity, the remaining cases can be eliminated.

We show that pairs exist in the remaining cases, and so eliminating the remaining cases with "a bit more ingenuity" is probably not achievable—at the very least, any argument required to eliminate the remaining cases would need to be more sophisticated in having to rely on the existence of a third square.

In order to derive a concise and effective SAT encoding for our search we make use of a duality between orthogonal Latin squares and transversal representation pairs. Although such a duality has long been used in searches for Latin squares, we also give an explicit formulation of how this duality arises via a composition operation on Latin squares. We found this viewpoint useful when deriving our encoding and surprisingly we were not able to find it expressed in prior literature.

References

1. Abel, R.J.R., Colbourn, C.J., Dinitz, J.H.: Mutually orthogonal Latin squares (MOLS). In: Handbook of Combinatorial Designs, pp. 186–218. Chapman and Hall/CRC (2006)
2. Appa, G., Magos, D., Mourtos, I.: Searching for mutually orthogonal Latin squares via integer and constraint programming. European Journal of Operational Research **173**(2), 519–530 (Sep 2006). https://doi.org/10.1016/j.ejor.2005.01.048
3. Appa, G., Mourtos, I., Magos, D.: Integrating Constraint and Integer Programming for the Orthogonal Latin Squares Problem, p. 17–32. Springer Berlin Heidelberg (2002). https://doi.org/10.1007/3-540-46135-3_2
4. Bailleux, O., Boufkhad, Y.: Efficient CNF Encoding of Boolean Cardinality Constraints, p. 108–122. Springer Berlin Heidelberg (2003). https://doi.org/10.1007/978-3-540-45193-8_8, http://dx.doi.org/10.1007/978-3-540-45193-8_8
5. Biere, A., Fleury, M.: Gimsatul, IsaSAT and Kissat entering the SAT competition 2022. Proc. of SAT Competition: Solver and Benchmark Descriptions, 2022 pp. 10–11 (2022)
6. Bose, R.C., Shrikhande, S.S.: On the falsity of Euler's conjecture about the non-existence of two orthogonal latin squares of order $4t+2$. Proceedings of the National Academy of Sciences **45**(5), 734–737 (May 1959). https://doi.org/10.1073/pnas.45.5.734
7. Bose, R.C., Shrikhande, S.S., Parker, E.T.: Further results on the construction of mutually orthogonal Latin squares and the falsity of Euler's conjecture. Canadian Journal of Mathematics **12**, 189–203 (1960). https://doi.org/10.4153/cjm-1960-016-5
8. Bose, R.C.: On the application of the properties of Galois fields to the problem of construction of Hyper-Græco-Latin squares. Sankhyā: The Indian Journal of Statistics **3**(4), 323–338 (1938), http://www.jstor.org/stable/40383859
9. Bright, C., Cheung, K.K.H., Stevens, B., Kotsireas, I., Ganesh, V.: A SAT-based resolution of Lam's problem. Proceedings of the AAAI Conference on Artificial Intelligence **35**(5), 3669–3676 (May 2021). https://doi.org/10.1609/aaai.v35i5.16483

10. Bright, C., Gerhard, J., Kotsireas, I., Ganesh, V.: Effective Problem Solving Using SAT Solvers, p. 205–219. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-41258-6_15
11. Bruck, R.H.: Finite nets. II. Uniqueness and imbedding. Pacific Journal of Mathematics **13**(2), 421–457 (Jun 1963). https://doi.org/10.2140/pjm.1963.13.421
12. Delisle, E.: The Search for a Triple of Mutually Orthogonal Latin Squares of Order Ten: Looking Through Pairs of Dimension Thirty-Five and Less. Master's thesis, University of Victoria (2010)
13. Egan, J., Wanless, I.M.: Enumeration of MOLS of small order. Mathematics of Computation **85**(298), 799–824 (Jul 2015). https://doi.org/10.1090/mcom/3010
14. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer, p. 228–245. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-40970-2_15
15. Huang, P., Liu, M., Ge, C., Ma, F., Zhang, J.: Investigating the existence of orthogonal golf designs via satisfiability testing. In: Proceedings of the 2019 International Symposium on Symbolic and Algebraic Computation. pp. 203–210. ISSAC '19, ACM (Jul 2019). https://doi.org/10.1145/3326229.3326232
16. Jin, J., Lv, Y., Ge, C., Ma, F., Zhang, J.: Investigating the Existence of Costas Latin Squares via Satisfiability Testing, p. 270–279. Springer International Publishing (2021). https://doi.org/10.1007/978-3-030-80223-3_19
17. Keedwell, A.D., Dénes, J.: Latin Squares and their Applications, Second Edition. Elsevier (2015). https://doi.org/10.1016/c2014-0-03412-0
18. Lam, C.W.H., Thiel, L., Swiercz, S.: The non-existence of finite projective planes of order 10. Canadian Journal of Mathematics **41**(6), 1117–1123 (Dec 1989). https://doi.org/10.4153/cjm-1989-049-4
19. Laywine, C.F., Mullen, G.L.: Discrete Mathematics Using Latin Squares, vol. 49. John Wiley & Sons (1998)
20. Lu, R., Liu, S., Zhang, J.: Searching for Doubly Self-orthogonal Latin Squares, p. 538–545. Springer Berlin Heidelberg (2011). https://doi.org/10.1007/978-3-642-23786-7_41
21. Ma, F., Zhang, J.: Finding orthogonal Latin squares using finite model searching tools. Science China Information Sciences **56**(3), 1–9 (Sep 2011). https://doi.org/10.1007/s11432-011-4343-3
22. Mann, H.B.: The construction of orthogonal Latin squares. The Annals of Mathematical Statistics **13**(4), 418–423 (1942)
23. Mann, H.B.: On orthogonal Latin squares. Bulletin of the American Mathematical Society **50**(4), 249–257 (1944). https://doi.org/10.1090/s0002-9904-1944-08127-5
24. Marques-Silva, J., Lynce, I.: Towards Robust CNF Encodings of Cardinality Constraints, p. 483–497. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-74970-7_35, http://dx.doi.org/10.1007/978-3-540-74970-7_35
25. McKay, B.D., Meynert, A., Myrvold, W.: Small Latin squares, quasigroups, and loops. Journal of Combinatorial Designs **15**(2), 98–119 (Mar 2006). https://doi.org/10.1002/jcd.20105
26. McKay, B.D., Piperno, A.: Practical graph isomorphism, II. Journal of Symbolic Computation **60**, 94–112 (Jan 2014). https://doi.org/10.1016/j.jsc.2013.09.003
27. Moore, E.H.: Tactical memoranda I-III. American Journal of Mathematics **18**(3), 264 (Jul 1896). https://doi.org/10.2307/2369797

28. Myrvold, W.: Negative results for orthogonal triples of Latin squares of order 10. Journal of Combinatorial Mathematics and Combinatorial Computing **29**, 95–106 (1999)
29. Parker, E.T.: Orthogonal Latin squares. Proceedings of the National Academy of Sciences **45**(6), 859–862 (Jun 1959). `https://doi.org/10.1073/pnas.45.6.859`
30. Parker, E.: On orthogonal Latin squares. 1960 Institute on Finite Groups **6**, 43–36 (1962). `https://doi.org/10.1090/pspum/006/0132704`
31. Roy, D.J.: Confirmation of the Non-existence of a Projective Plane of Order 10. Master's thesis, Carleton University (2011)
32. Rubin, N., Bright, C., Cheung, K., Stevens, B.: Improving integer and constraint programming for Graeco–Latin squares. In: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI). IEEE (Nov 2021). `https://doi.org/10.1109/ictai52525.2021.00096`
33. Tarry, G.: Le problème des 36 officiers. C. R. Assoc. Française Avancement Sci., Paris 29, No. 2, 170-203 (1900). (1900)
34. Vardi, M.Y.: Boolean satisfiability: theory and engineering. Communications of the ACM **57**(3), 5–5 (Mar 2014). `https://doi.org/10.1145/2578043`
35. Wanless, I.: Transversals in Latin Squares: A Survey, p. 403–437. Cambridge University Press (Jun 2011). `https://doi.org/10.1017/cbo9781139004114.010`
36. Zaikin, O., Kochemazov, S.: The search for systems of diagonal Latin squares using the SAT@home project. International Journal of Open Information Technologies **3**(11), 4–9 (2015)
37. Zhang, H.: Specifying Latin square problems in propositional logic. In: Automated reasoning and its applications: essays in honor of Larry Wos, pp. 115–146 (1997)
38. Zhang, H.: Combinatorial designs by SAT solvers. In: Handbook of Satisfiability. pp. 819–858. IOS Press (Feb 2021). `https://doi.org/10.3233/faia201005`

Appendix

In the appendix we provide eight explicit pairs we found which prove the existence of TRPs for Myrvold's eight unresolved cases [28].

type S

0	8	6	2	5	1	9	3	4	7
1	2	3	5	9	6	8	0	7	4
2	3	4	0	6	5	7	9	1	8
4	9	2	3	1	7	6	8	5	0
7	0	9	1	8	2	4	5	6	3
8	6	1	4	2	0	3	7	9	5
9	1	0	7	3	4	5	6	8	2
5	7	8	9	4	3	1	2	0	6
6	5	7	8	0	9	2	4	3	1
3	4	5	6	7	8	0	1	2	9

type X

3	1	9	2	6	7	8	4	0	5
4	2	7	1	5	3	0	6	9	8
8	3	2	7	0	6	1	5	4	9
9	4	3	0	8	1	2	7	5	6
0	7	1	6	9	5	4	8	3	2
1	8	5	4	3	9	7	2	6	0
2	9	6	8	4	0	5	1	7	3
6	0	8	5	1	4	3	9	2	7
7	5	0	9	2	8	6	3	1	4
5	6	4	3	7	2	9	0	8	1

type U

0	9	4	2	1	6	7	8	3	5
1	2	8	4	6	3	9	5	7	0
4	0	9	5	3	2	8	7	6	1
6	8	7	1	2	0	5	3	9	4
7	1	3	0	9	5	4	6	8	2
9	5	2	3	0	7	6	4	1	8
2	3	6	8	4	9	1	0	5	7
5	6	0	7	8	1	2	9	4	3
8	7	1	9	5	4	3	2	0	6
3	4	5	6	7	8	0	1	2	9

type X

0	8	1	4	3	5	6	9	2	7
1	0	7	3	8	6	4	2	5	9
2	4	9	7	1	3	5	6	0	8
3	5	8	1	9	2	7	0	4	6
4	1	6	9	0	8	2	3	7	5
5	7	2	0	6	9	8	1	3	4
6	3	0	5	7	4	9	8	1	2
7	9	5	8	2	1	3	4	6	0
8	6	3	2	4	7	0	5	9	1
9	2	4	6	5	0	1	7	8	3

type V

0	9	1	4	3	8	7	6	5	2
2	1	9	3	6	7	4	5	8	0
3	6	5	0	7	1	9	2	4	8
5	0	4	6	1	2	3	8	7	9
7	3	8	2	5	0	6	9	1	4
8	2	0	1	9	6	5	4	3	7
1	7	2	9	4	5	8	3	0	6
6	8	3	5	0	4	2	7	9	1
9	4	7	8	2	3	1	0	6	5
4	5	6	7	8	9	0	1	2	3

type X

1	3	5	4	2	6	0	7	8	9
3	9	0	2	6	4	8	1	7	5
7	1	4	9	0	3	5	6	2	8
9	8	2	0	1	7	6	4	5	3
0	4	9	7	5	1	2	8	3	6
2	5	3	6	4	8	9	0	1	7
4	0	8	3	9	5	7	2	6	1
5	6	7	1	8	0	4	3	9	2
6	2	1	8	7	9	3	5	0	4
8	7	6	5	3	2	1	9	4	0

type W

0	3	8	1	4	5	6	9	7	2
1	8	2	0	6	7	4	3	9	5
2	1	7	4	9	3	8	6	5	0
5	0	6	7	1	2	9	4	3	8
6	2	1	9	3	4	5	0	8	7
4	5	9	8	2	0	1	7	6	3
7	9	0	2	5	6	3	8	4	1
8	7	4	3	0	9	2	5	1	6
9	6	3	5	8	1	7	2	0	4
3	4	5	6	7	8	0	1	2	9

type X

3	6	8	9	0	2	4	7	5	1
5	4	7	0	2	1	3	9	8	6
6	8	9	2	1	3	0	5	7	4
8	2	3	7	9	0	6	1	4	5
0	9	4	5	3	7	1	6	2	8
1	5	0	6	4	9	8	2	3	7
2	7	5	1	6	4	9	8	0	3
4	1	6	3	5	8	7	0	9	2
7	0	1	4	8	5	2	3	6	9
9	3	2	8	7	6	5	4	1	0

type X

4	9	0	3	8	2	5	6	7	1
5	2	9	4	0	1	3	7	6	8
6	7	2	9	1	3	4	8	5	0
7	0	3	2	9	4	8	5	1	6
0	1	6	7	5	8	9	2	3	4
1	6	5	8	2	9	7	0	4	3
3	4	8	0	7	6	1	9	2	5
8	5	1	6	4	0	2	3	9	7
9	8	7	1	3	5	6	4	0	2
2	3	4	5	6	7	0	1	8	9

type X

1	7	9	5	3	0	8	6	2	4
2	4	1	9	0	8	6	5	7	3
6	2	3	0	8	5	9	1	4	7
7	1	4	6	2	3	5	9	0	8
0	5	8	3	6	9	4	7	1	2
3	9	7	8	4	1	0	2	5	6
5	6	0	1	7	4	2	8	3	9
8	3	5	7	9	2	1	4	6	0
9	0	2	4	5	6	7	3	8	1
4	8	6	2	1	7	3	0	9	5

type U

1	2	0	8	6	7	9	4	3	5
2	5	9	1	8	0	4	7	6	3
4	0	6	7	2	3	1	5	9	8
5	7	2	9	3	1	6	0	8	4
6	1	3	5	9	2	7	8	4	0
9	3	8	0	1	4	5	2	7	6
0	8	7	2	4	6	3	9	5	1
8	9	4	3	0	5	2	6	1	7
7	6	1	4	5	9	8	3	0	2
3	4	5	6	7	8	0	1	2	9

type U

0	4	8	5	2	1	9	3	6	7
1	0	3	9	4	5	8	7	2	6
2	3	6	8	0	9	7	1	5	4
3	1	4	7	6	0	5	9	8	2
6	8	2	3	1	7	4	5	0	9
9	7	1	2	8	3	0	6	4	5
5	6	9	0	7	2	3	4	1	8
7	5	0	6	3	4	2	8	9	1
4	9	7	1	5	8	6	2	3	0
8	2	5	4	9	6	1	0	7	3

type U

0	7	1	2	8	5	6	4	9	3
1	6	0	8	3	4	9	5	7	2
4	9	2	3	1	6	8	7	0	5
5	0	8	1	9	2	4	3	6	7
6	1	9	7	2	3	5	8	4	0
7	2	3	5	4	0	1	9	8	6
8	3	6	4	0	9	7	2	5	1
9	8	7	0	5	1	2	6	3	4
2	5	4	9	6	7	3	0	1	8
3	4	5	6	7	8	0	1	2	9

type W

0	4	2	1	5	9	3	8	7	6
1	3	4	0	8	6	5	9	2	7
2	1	8	5	0	4	6	7	3	9
3	2	9	8	1	5	7	0	6	4
6	9	7	2	3	0	4	1	5	8
4	0	3	7	6	8	2	5	9	1
7	5	0	6	9	1	8	2	4	3
8	7	5	3	4	2	9	6	1	0
9	6	1	4	2	7	0	3	8	5
5	8	6	9	7	3	1	4	0	2

type W

2	8	7	9	3	0	5	6	4	1
4	2	1	8	0	6	9	3	7	5
5	1	0	7	4	2	8	9	3	6
7	0	3	5	6	1	2	4	9	8
9	3	6	2	1	5	4	7	8	0
0	7	9	4	8	3	1	5	6	2
1	6	8	0	9	4	3	2	5	7
6	5	4	1	2	9	7	8	0	3
8	9	2	3	5	7	6	0	1	4
3	4	5	6	7	8	0	1	2	9

type W

0	6	4	2	3	7	8	1	9	5
1	4	2	7	6	0	9	5	8	3
3	8	0	1	9	5	6	4	7	2
5	0	1	3	8	9	4	6	2	7
7	2	6	0	4	3	5	8	1	9
2	3	8	5	7	6	1	9	0	4
4	1	3	9	5	8	7	2	6	0
6	9	7	4	1	2	0	3	5	8
9	7	5	8	2	1	3	0	4	6
8	5	9	6	0	4	2	7	3	1