Enumerating Extended Self-orthogonal Diagonal Latin Squares of Order up to 10

Oleg Zaikin
ISDCT SB RAS
Lermontov street 134
664033 Irkutsk
Russia
oleg.zaikin@icc.ru

Eduard Vatutin
Southwest State University
50 let Oktyabrya 94
305040 Kursk
Russia
evatutin@rambler.ru

Curtis Bright
University of Waterloo
200 University Ave. W
Waterloo, Ontario
Canada
cbright@uwaterloo.ca

Abstract

We enumerate all extended self-orthogonal diagonal Latin squares of order up to 10. Our method reduces the problem of enumerating extended self-orthogonal diagonal Latin squares to a satisfiability (SAT) problem, and we find all solutions of the SAT problem using a SAT solver. We additionally show that there is no triple of mutually orthogonal diagonal Latin squares of order 10 containing an extended self-orthogonal diagonal Latin square.

1 Introduction

A *Latin square* $A = (a_{ij})$ of order n is an $n \times n$ array such that each row and each column contains all symbols from the set $\{0, 1, \ldots, n-1\}$ [21]. Two Latin squares A and B of the same order are *isotopic* if B can be formed by permuting rows, columns, and symbols of A. An *isotopy class* of Latin squares is an equivalence class for the isotopy relation. If a Latin square $A = (a_{ij})$ of order n is represented as an orthogonal array $\{\,(i, j, a_{ij}) : i, j = 0, 1, \ldots, n-1\,\}$, permuting columns of the orthogonal array produces $3! = 6$ new Latin squares called

parastrophes (or *conjugates*) of the original Latin square A. The $(1, 2, 3)$-parastrophe of a Latin square is the Latin square itself. The $(2, 1, 3)$-parastrophe is the transpose of that square because it is produced by switching rows and columns. Similarly, the $(1, 3, 2)$-parastrophe is obtained by switching columns and symbols. A *parastrophy class* of Latin squares is an equivalence class for the parastrophy relation. The set of all parastrophes from an isotopy class form a *main class*, while two Latin squares from the same main class are called *paratopic*.

A *diagonal Latin square* is a Latin square whose main diagonal and main anti-diagonal have distinct entries. Isotopy and main classes of diagonal Latin squares are much smaller than that of ordinary Latin squares, because the diagonality condition restricts most transformations used to form equivalent Latin squares. Given a diagonal Latin square, combinations of the following transformations are applied to construct its main class of diagonal Latin squares: symbol permutations; rotations (clockwise or anticlockwise); reflections (horizontal, vertical, across the main diagonal, and across the main anti-diagonal); and M-transformations. There are two types of M-transformations (cf. [24]): (1) two columns positioned symmetrically with respect to the middle are permuted, then two rows with the same indices are permuted; and (2) at least two columns in the left half of a square are permuted, then the corresponding symmetric columns in the right part are permuted in the same way, and finally the permutation is done for rows with the same indices.

Two Latin squares $A = (a_{ij})$ and $B = (b_{ij})$ of the same order are *orthogonal* if all ordered pairs (a_{ij}, b_{ij}) are distinct. A set of Latin squares of the same order n that are pairwise orthogonal is called a set of *mutually orthogonal Latin squares* (*MOLS*) of order n. For diagonal Latin squares, *MODLS* is defined similarly. The first pair of MODLS of order 10 was found by Brown et al. in 1993 [8].

If A and its transpose A^T are orthogonal Latin squares then A is called a *self-orthogonal Latin square* (*SOLS*) [16]. A *self-orthogonal diagonal Latin square* (*SODLS*) is defined similarly [2]. Vatutin and Belyshev [36] proposed a generalization of SODLSs called *extended self-orthogonal diagonal Latin square* (*ESODLS*) and enumerated ESODLSs of order up to 8. The motivation behind introducing ESODLSs was twofold: (1) to master a new approach for generating pairs of MODLS; and (2) to check if there exists a triple of MODLS of order 10 such that at least one square is an ESODLS. The present study is aimed at enumerating ESODLSs of order up to 10.

We formally describe ESODLSs in Section 2. We propose an enumeration algorithm aimed at ESODLSs in Section 3, explain how the most time-consuming part of the algorithm is reduced to finding all solutions of a satisfiability problem in Section 4, and we present the results of our computational experiments for orders 9 and 10 in Section 5. Finally, we discuss related studies and present the conclusions.

2 Extended self-orthogonal Latin squares

An *extended self-orthogonal diagonal Latin square* (*ESODLS*) is a diagonal Latin square that has an orthogonal mate in its main class of diagonal Latin squares [36]. In the rest of the paper, only main classes of diagonal Latin squares are discussed. ESODLSs are a generalization of

SODLSs, since the transpose of a diagonal Latin square is the $(2, 1, 3)$-parastrophe of that Latin square and hence is in its main class.

Consider Latin squares $A = (a_{ij})$ and $B = (b_{ij})$ of order n. A *cells mapping scheme* (*CMS*) for the ordered pair (A, B) is an $n \times n$ array $C = (c_{ij})$ filled with integer numbers from $\{0, \ldots, n^2 - 1\}$ such that all entries of C are distinct and $a_{ij} = b_{i_2 j_2}$ where $i_2 = \lfloor c_{ij}/n \rfloor$ and $j_2 = c_{ij} \bmod n$ [39]. The *trivial* CMS of order n maps a Latin square onto itself, i.e., $c_{ij} = i \cdot n + j$, so $(i_2, j_2) = (i, j)$.

Below two examples of diagonal Latin squares of order 7 are presented. The first one is a SODLS, as it is orthogonal to its transpose. The second diagonal Latin square is not a SODLS, but it is an ESODLS since it is orthogonal to the square formed by reflecting its entries across its middle column. Thus, the second diagonal Latin square is isotopic (and, therefore, paratopic) to one of its orthogonal mates. For each square, a nontrivial CMS, and the corresponding orthogonal mate are also shown.

Example 1. A SODLS (and, therefore, an ESODLS) of order 7, the CMS for the transposition operation, and the corresponding paratopic orthogonal mate.

$$\begin{bmatrix} 0 & 2 & 4 & 6 & 5 & 3 & 1 \\ 6 & 1 & 0 & 5 & 3 & 2 & 4 \\ 1 & 5 & 2 & 4 & 0 & 6 & 3 \\ 4 & 6 & 5 & 3 & 1 & 0 & 2 \\ 3 & 0 & 6 & 2 & 4 & 1 & 5 \\ 2 & 4 & 3 & 1 & 6 & 5 & 0 \\ 5 & 3 & 1 & 0 & 2 & 4 & 6 \end{bmatrix} \begin{bmatrix} 0 & 7 & 14 & 21 & 28 & 35 & 42 \\ 1 & 8 & 15 & 22 & 29 & 36 & 43 \\ 2 & 9 & 16 & 23 & 30 & 37 & 44 \\ 3 & 10 & 17 & 24 & 31 & 38 & 45 \\ 4 & 11 & 18 & 25 & 32 & 39 & 46 \\ 5 & 12 & 19 & 26 & 33 & 40 & 47 \\ 6 & 13 & 20 & 27 & 34 & 41 & 48 \end{bmatrix} \begin{bmatrix} 0 & 6 & 1 & 4 & 3 & 2 & 5 \\ 2 & 1 & 5 & 6 & 0 & 4 & 3 \\ 4 & 0 & 2 & 5 & 6 & 3 & 1 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 5 & 3 & 0 & 1 & 4 & 6 & 2 \\ 3 & 2 & 6 & 0 & 1 & 5 & 4 \\ 1 & 4 & 3 & 2 & 5 & 0 & 6 \end{bmatrix}.$$

Example 2. An ESODLS of order 7 that is not a SODLS, a nontrivial CMS, and the corresponding paratopic orthogonal mate.

$$\begin{bmatrix} 0 & 6 & 4 & 5 & 2 & 3 & 1 \\ 6 & 1 & 5 & 4 & 3 & 2 & 0 \\ 4 & 5 & 2 & 6 & 0 & 1 & 3 \\ 1 & 2 & 0 & 3 & 5 & 6 & 4 \\ 2 & 3 & 6 & 1 & 4 & 0 & 5 \\ 3 & 4 & 1 & 0 & 6 & 5 & 2 \\ 5 & 0 & 3 & 2 & 1 & 4 & 6 \end{bmatrix} \begin{bmatrix} 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 13 & 12 & 11 & 10 & 9 & 8 & 7 \\ 20 & 19 & 18 & 17 & 16 & 15 & 14 \\ 27 & 26 & 25 & 24 & 23 & 22 & 21 \\ 34 & 33 & 32 & 31 & 30 & 29 & 28 \\ 41 & 40 & 39 & 38 & 37 & 36 & 35 \\ 48 & 47 & 46 & 45 & 44 & 43 & 42 \end{bmatrix} \begin{bmatrix} 1 & 3 & 2 & 5 & 4 & 6 & 0 \\ 0 & 2 & 3 & 4 & 5 & 1 & 6 \\ 3 & 1 & 0 & 6 & 2 & 5 & 4 \\ 4 & 6 & 5 & 3 & 0 & 2 & 1 \\ 5 & 0 & 4 & 1 & 6 & 3 & 2 \\ 2 & 5 & 6 & 0 & 1 & 4 & 3 \\ 6 & 4 & 1 & 2 & 3 & 0 & 5 \end{bmatrix}.$$

By applying to the trivial CMS of order n all possible combinations of rotations, reflections, and M-transformations, a set of *ESODLS CMSs* is obtained [39]. This set contains all possible CMSs that match ESODLSs of order n and their paratopic orthogonal mates. It is clear that both CMSs of order 7 presented above are ESODLS CMSs. We enumerated all ESODLS CMSs of order up to 10. As a result, we obtained the following sequence:

$$1, 0, 0, 2, 4, 96, 192, 1536, 1536, 15360.$$

This is sequence A299784 in the On-Line Encyclopedia of Integer Sequences (OEIS) [34]. The sequence represents the maximal size of a main class of diagonal Latin squares of order n with a fixed first row.

3 Enumeration algorithm

In this section we introduce two enumeration algorithms aimed at ESODLSs. The second algorithm is an improved version of the first one. Without loss of generality, we suppose the first row of the ESODLS is in ascending order, i.e., $0, \ldots, n-1$.

We propose algorithm A as follows.

1. Generate all ESODLS CMSs of order n.
2. For each ESODLS CMS, find all pairs of MODLS of order n such that a pair matches the CMS and the first row of the first diagonal Latin square from the pair is in ascending order, i.e., $0, \ldots, n-1$.
3. Collect distinct first diagonal Latin squares from the pairs of MODLS found in step 2.

It is clear that, by definition of ESODLS CMSs, algorithm A enumerates ESODLSs of order n with the first row in ascending order.

A *partial diagonal Latin square* $A = (a_{ij})$ of order n is an $n \times n$ array filled with symbols from the set $\{0, \ldots, n-1\}$ in such a way that in each row, each column, main diagonal, and main anti-diagonal all entries are distinct [21]. An *X-based partial diagonal Latin square* is a partial diagonal Latin square such that its main diagonal and main anti-diagonal are completely filled, while the remaining cells are not filled [37]. The X-based partial diagonal Latin square of order 7 that corresponds to the ESODLS from Examples 1 and 2 is presented in Example 3.

Example 3. An X-based partial diagonal Latin square of order 7.

$$\begin{bmatrix} 0 & - & - & - & - & - & 1 \\ - & 1 & - & - & - & 2 & - \\ - & - & 2 & - & 0 & - & - \\ - & - & - & 3 & - & - & - \\ - & - & 6 & - & 4 & - & - \\ - & 4 & - & - & - & 5 & - \\ 5 & - & - & - & - & - & 6 \end{bmatrix}.$$

Equivalence classes of X-based partial diagonal Latin squares of order n can be formed as follows [37]:

1. Generate all ESODLS CMSs of order n.
2. Generate all X-based partial diagonal Latin squares of order n whose main diagonal has entries $0, \ldots, n-1$.
3. Apply all ESODLS CMSs of order n to each X-based partial diagonal Latin square generated in step 2, thus producing a set of X-based partial diagonal Latin squares. Note that every ESODLS CMS transforms an X-based partial diagonal Latin square to an X-based partial diagonal Latin square.

4. In each partial X-based diagonal Latin square obtained in step 3, permute the symbols to have the main diagonal $0, \ldots, n-1$. As a result, all distinct X-based partial diagonal Latin squares from the same equivalence class are formed.

5. Choose the lexicographically minimal representative from the class formed in step 4. Since the main diagonal is the same, only the main anti-diagonal is taken into account.

6. Collect distinct lexicographically minimal representatives obtained in steps 3–5.

There are 3, 20, 20, and 67 equivalence classes of X-based partial diagonal Latin squares of order 7, 8, 9, and 10, respectively—see sequence A309283 in the OEIS [34].

We propose algorithm B as an improved version of algorithm A to enumerate ESODLSs of order n having first row in ascending order.

1. Generate all ESODLS CMSs of order n, and all lexicographically minimal representatives of equivalence classes of X-based partial diagonal Latin squares of order n as above.

2. For each pair (ESODLS CMS, lexicographically minimal X-based representative), find all pairs of MODLS of order n such that (1) the pair matches the CMS and (2) the first diagonal Latin square in the pair matches the X-based partial diagonal Latin square.

3. Collect distinct first diagonal Latin squares (i.e., ESODLSs) from the pairs of MODLS found in step 2.

4. Apply to each ESODLS collected in step 3 all ESODLS CMSs and choose all distinct ESODLSs with the first row in ascending order.

Note that as a result of step 3 all ESODLSs of order n are enumerated matching lexicographically minimal representatives of X-based partial diagonal Latin squares of order n. Main diagonals of these ESODLSs are in ascending order because they were built from X-based partial diagonal Latin squares having that property. Then for each ESODLS collected in step 3 its main class of diagonal Latin squares (ESODLSs) is produced in step 4 by applying all ESODLS CMSs. In most ESODLSs from these main classes the main diagonals will not be in ascending order since most corresponding transformations break this condition. Only ESODLSs with the first row in ascending order are chosen from the main classes in step 4. Since only lexicographically minimal X-based fillings are used in algorithm B, a significant reduction in the search space is achieved when contrasted with algorithm A which did not do this symmetry breaking.

4 SAT encoding

Step 2 of algorithm B from the previous section (enumerating all pairs of MODLS of order n satisfying certain constraints) is the most time-consuming part of our approach. In order to enumerate pairs of MODLS of order n as efficiently as possible, we use a satisfiability

(SAT) approach. In particular, we encode the existence of MODLS in a SAT formulation and find all pairs of MODLS via an AllSAT solver. This section gives preliminaries on SAT and describes the SAT encoding.

4.1 Preliminaries on SAT

A propositional Boolean formula is *satisfiable* if it has a satisfying assignment; otherwise it is *unsatisfiable*. A *satisfying assignment* is a truth assignment to the Boolean variables in the formula that makes the formula true. The *Boolean satisfiability problem* (SAT) is to determine whether a given propositional Boolean formula is satisfiable [3]. SAT was historically the first problem shown to be NP-complete [10]. The *AllSAT* problem is to enumerate all satisfying assignments for a given propositional Boolean formula. A propositional Boolean formula is in Conjunctive Normal Form (*CNF*) if it is a conjunction of clauses. A *clause* is a disjunction of literals, where a *literal* is a Boolean variable or its negation. A *unit clause* is a clause containing exactly one literal. For convenience, we may write the clause $\neg x \vee \neg y \vee z$ as $(x \wedge y) \rightarrow z$.

4.2 Encoding

Consider two diagonal Latin squares A and B of order n. For a given ESODLS CMS and an X-based partial diagonal Latin square of order n, our goal is to construct a CNF formula so that solving the AllSAT problem for the formula results in all pairs of MODLS of order n needed in our algorithm—namely, having entries corresponding to a specific CMS and whose diagonal and anti-diagonal entries match a specific X-based filling. Note that the main diagonal and main anti-diagonal of both A and B are known when an ESODLS CMS and X-based filling are applied simultaneously.

A Latin square of order n can be represented as an $n \times n \times n$ Boolean array where the dimensions are associated with rows, columns, and symbols [38, 5]. In the array, the cell (i, j, k) contains 1 if the Latin square's cell (i, j) contains k, otherwise (i, j, k) contains 0. Introduce n^3 Boolean variables $x(i, j, k)$ (with $i, j, k \in \{0, \ldots, n-1\}$) to encode the $n \times n \times n$ array. The following *Exactly One* (EO) constraints ensure that the array corresponds to a diagonal Latin square:

1. $\mathrm{EO}(x(i, j, 0), \ldots, x(i, j, n-1))$ for $i,\ j = 0, \ldots, n-1$;
2. $\mathrm{EO}(x(i, 0, k), \ldots, x(i, n-1, k))$ for $i,\ k = 0, \ldots, n-1$;
3. $\mathrm{EO}(x(0, j, k), \ldots, x(n-1, j, k))$ for $j,\ k = 0, \ldots, n-1$;
4. $\mathrm{EO}(x(0, 0, k), \ldots, x(n-1, n-1, k))$ for $k = 0, \ldots, n-1$;
5. $\mathrm{EO}(x(n-1, 0, k), \ldots, x(0, n-1, k))$ for $k = 0, \ldots, n-1$.

In these constraints $\mathrm{EO}(x_1, \ldots, x_m) = 1$ if and only if exactly one Boolean variable among $x_1, \ldots, x_m$ is assigned true and all remaining variables are assigned false. The first constraint ensures that each Latin square's cell contains exactly one symbol $0, \ldots, n-1$. The second/third constraint ensures that each row/column contains each symbol $0, \ldots, n-1$ exactly once. The forth/fifth constraint ensures that the main diagonal/anti-diagonal contains each symbol $0, \ldots, n-1$ exactly once.

The EO constraint for m Boolean variables $x_1, \ldots, x_m$ is encoded to SAT by the clause $(x_1 \vee \cdots \vee x_m)$ and the $m(m-1)/2$ clauses $(\neg x_i \vee \neg x_j)$ for $1 \leq i < j \leq m$. The first clause encodes the at-least-one predicate, while the remaining clauses encode the at-most-one predicate. The latter can be encoded in several ways [30] and we chose the pairwise encoding to avoid introducing new auxiliary variables. In total, $3n^2 + 2n$ EOs are needed for a diagonal Latin square of order n. This means that the search for a diagonal Latin square of order n is encoded to SAT by n^3 Boolean variables and $(3n^2+2n)(n(n-1)/2+1) = 1.5n^4 - 0.5n^3 + 2n^2 + 2n$ clauses. The search for a Latin square is encoded by $1.5n^4 - 1.5n^3 + 3n^2$ clauses since only $3n^2$ EOs are needed which correspond to the first three constraints.

To encode the orthogonality condition, a SAT formulation of the CP-index encoding from Rubin et al. [33] is used. To describe the CP-index encoding, we first describe the product of two Latin squares. Suppose A and B are two Latin squares of order n and denote the ith row of A by A_i (and similarly for B). The row A_i can equivalently be considered as a permutation of the row $(0, \ldots, n-1)$. Then the *product* square AB is the square whose ith row forms the permutation $A_i \circ B_i$ (with the composition performed right-to-left) and the *inverse* square A^{-1} is the square whose ith row forms the permutation A_i^{-1}. For example, if

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \\ 1 & 0 & 3 & 2 \end{bmatrix}, \quad \text{then} \quad \begin{aligned} A_0 &= B_0 = (0), \\ A_1 &= B_3 = (01)(23), \\ A_2 &= B_1 = (02)(13), \\ A_3 &= B_2 = (03)(12). \end{aligned}$$

Furthermore, A and B are both self-inverses and

$$AB = AB^{-1} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \end{bmatrix}.$$

Note that the product of two Latin squares is not in general a Latin square, although in this example the product square AB *is* in fact a Latin square.

The CP-index orthogonality encoding is based on the following theorem of Mann [28] that guarantees the existence of a product Latin square "witnessing" the orthogonality of two orthogonal Latin squares. For example, because the product AB^{-1} in the above example is a Latin square, Mann's theorem implies that the Latin squares A and B used to form the product are orthogonal.

Theorem 4 ([28])**.** *Two Latin squares A and B are orthogonal if and only if AB^{-1} is a Latin square.*

To use this theorem, we form the new square $Z \coloneqq AB^{-1}$ and assert that it is a Latin square. This entails introducing new variables $z(i,j,k)$ for $i,j,k \in \{0,\ldots,n-1\}$ denoting that the (i,j)th entry of the square Z contains k. Asserting Z is a Latin square uses the encoding already described. To encode $ZB = A$, we encode that the b_{ij}th entry of Z_i is equal to the jth entry of A_i for all $0 \leq i,j < n$. In a constraint programming solver, this can natively be encoded via an indexing constraint $Z_i[B_i[j]] = A_i[j]$ (i.e., $z_{i,b_{ij}} = a_{ij}$). To encode this in a SAT context, we rewrite this in the equivalent form

$$b_{ij} = l \leftrightarrow z_{il} = a_{ij} \qquad \text{for all } 0 \leq i,j,l < n.$$

From this, we derive formulae of the form

$$\begin{aligned}(b_{ij} = l \wedge z_{il} = k) &\rightarrow a_{ij} = k,\\ (b_{ij} = l \wedge a_{ij} = k) &\rightarrow z_{il} = k, \text{ and}\\ (z_{il} = k \wedge a_{ij} = k) &\rightarrow b_{ij} = l\end{aligned}$$

for $0 \leq i,j,k,l < n$. These correspond to $3n^4$ clauses $(b(i,j,l) \wedge z(i,l,k)) \rightarrow a(i,j,k)$, $(b(i,j,l) \wedge a(i,j,k)) \rightarrow z(i,l,k)$, and $(z(i,l,k) \wedge a(i,j,k)) \rightarrow b(i,j,l)$ where each index variable ranges over $0,\ldots,n-1$. Not all of these clauses are strictly necessary; intuitively, once all the entries in two of the three squares A, B, and Z are known, the entries in the remaining square are completely determined via one of the relationships $A = ZB$, $Z = AB^{-1}$, or $B = Z^{-1}A$, and Theorem 4 implies that A and B are orthogonal. Thus, it would be sufficient to use only n^4 clauses (those from just one of the three kinds of clauses relating A, B, and Z). However, we include all $3n^4$ clauses in our encoding as the redundancy helps the SAT solver perform propagations earlier.

Since we have to encode two diagonal Latin squares, one Latin square, and the orthogonality condition, so far there are $3n^3$ Boolean variables and $7.5n^4 - 2.5n^3 + 7n^2 + 4n$ clauses in the CNF. This CNF encodes the search for pairs of MODLS of order n. The X-based filling is enforced on the first diagonal Latin square A by adding $2n$ unit clauses if n is even (while if n is odd $2n-1$ unit clauses are added, because in this case the center entry appears on both the diagonal and anti-diagonal). Each such unit clause encodes that a cell in the main diagonal or main anti-diagonal has the value specified by the X-based filling. Finally, the ESODLS CMS is applied. Suppose that according to the CMS $a_{ij} = b_{i_2 j_2}$ and a_{ij} is encoded by Boolean variables $x_{k_1},\ldots,x_{k_n}$, while $b_{i_2 j_2}$ is encoded by $x_{l_1},\ldots,x_{l_n}$. Then $x_{k_i} \leftrightarrow x_{l_i}$ is encoded via the $2n$ binary clauses $(x_{k_i} \vee \neg x_{l_i}) \wedge (\neg x_{k_i} \vee x_{l_i})$ for $i = 1,\ldots,n$. In total, the CMS is encoded by adding $2n^3$ binary clauses. Now the CNF encodes the search for pairs of MODLS of order n where the pair matches the ESODLS CMS and the first diagonal Latin square in the pair matches the X-based filling.

Note that EO constraints for the main diagonal/anti-diagonal of the first diagonal Latin square are redundant since the corresponding entries are known and distinct due to the X-based filling. The same holds for the second diagonal Latin square due to the ESODLS CMS and the X-based filling. Therefore, the clauses that encode these four EO constraints can be safely excluded. However, we did not exclude them because a state-of-the-art SAT solver does it by itself in a fraction of a second during preprocessing by applying the unit propagation rule [11].

5 Computational experiments

We applied the proposed algorithm to enumerate all ESODLSs of order up to 10 with the first row in ascending order. As a result, we obtained the sequence counting the number of ESODLSs of order n:

$$1, 0, 0, 2, 4, 0, 256, 4608, 24437088, 510566400.$$

By multiplying these numbers by $n!$, we obtained the sequence representing the number of ESODLSs of order up to 10:

$$1, 0, 0, 48, 480, 0, 1290240, 185794560, 8867730493440, 1852743352320000.$$

These are sequences A309598 and A309599 in the OEIS. Currently, the counts for ESODLSs of order up to 9 are in the OEIS and we are adding the numbers for order 10 upon publication of the present paper.

By applying ESODLS CMSs to ESODLSs with the first row in ascending order, all main classes of ESODLSs were formed. The number of these main classes is the sequence

$$1, 0, 0, 1, 1, 0, 5, 23, 18865, 33240.$$

This is sequence A309210 in the OEIS, which currently contains entries up to order 9 and we will be adding the number for order 10. Note that the existing numbers in these three sequences were obtained by an exhaustive search. Since they coincide with our numbers, it serves as a double-check on the correctness of our AllSAT-based enumeration algorithm. Also, our numbers in order 10 coincide with the lower bounds published in [36].

5.1 ESODLS of order 9

The proposed algorithm and the SAT encoding were implemented in C++ and Python; the sources are available online [42].

For order 9, there are 20 equivalence classes of X-based partial diagonal Latin squares and 1 536 ESODLS CMSs. Therefore, 30 720 CNFs were constructed in step 2 of algorithm B. We applied the following AllSAT solvers to these CNFs:

- CLASP [13] version 3.3.10;
- CRYPTOMINISAT [35] version 5.11.21.

CLASP can be run in the following configurations: auto, crafty, frumpy, handy, jumpy, many, trendy, and tweety. All these configurations were tried. The experiments were held on a personal computer equipped with a 16-core CPU AMD Ryzen 9 3950X. All CPU cores were employed by running copies of a solver on CNFs in parallel. In Table 1, the wall-clock runtimes are presented.

According to the table, CLASP-CRAFTY is the fastest solver, so for order 10 this solver was chosen. In these experiments, 163 198 pairs of MODLS were found. It turned out that all first ESODLSs from the pairs are distinct.

Solver	Runtime
CLASP-AUTO	30 min
CLASP-CRAFTY	19 min
CLASP-FRUMPY	23 min
CLASP-HANDY	1 h 10 min
CLASP-JUMPY	32 min
CLASP-MANY	31 min
CLASP-TRENDY	31 min
CLASP-TWEETY	38 min
CRYPTOMINISAT	2 h 15 min

Table 1: Wall-clock runtimes of AllSAT solvers when enumerating ESODLSs of order 9.

5.2 ESODLS of order 10

For order 10, there are 67 equivalence classes of X-based partial diagonal Latin squares and 15 360 ESODLS CMSs. Therefore, 1 029 120 CNFs were constructed in the second step of algorithm B from Section 3. The experiments were performed on the supercomputer "Akademik V.M. Matrosov" [31]. 15 of the supercomputer's nodes were used and each node was equipped with two 16-core AMD Opteron 6276 CPUs. The CNFs corresponding to ESODLS CMSs number 1 and 14 881 (numbering from 0) turned out to be much harder than CNFs of the remaining ESODLS CMSs. These two CMSs are presented below. Note that ESODLS CMS number 1 corresponds to the transposition that produces SOLSs and SODLSs.

Example 5. ESODLS CMS number 1.

$$\begin{bmatrix}
0 & 10 & 20 & 30 & 40 & 50 & 60 & 70 & 80 & 90 \\
1 & 11 & 21 & 31 & 41 & 51 & 61 & 71 & 81 & 91 \\
2 & 12 & 22 & 32 & 42 & 52 & 62 & 72 & 82 & 92 \\
3 & 13 & 23 & 33 & 43 & 53 & 63 & 73 & 83 & 93 \\
4 & 14 & 24 & 34 & 44 & 54 & 64 & 74 & 84 & 94 \\
5 & 15 & 25 & 35 & 45 & 55 & 65 & 75 & 85 & 95 \\
6 & 16 & 26 & 36 & 46 & 56 & 66 & 76 & 86 & 96 \\
7 & 17 & 27 & 37 & 47 & 57 & 67 & 77 & 87 & 97 \\
8 & 18 & 28 & 38 & 48 & 58 & 68 & 78 & 88 & 98 \\
9 & 19 & 29 & 39 & 49 & 59 & 69 & 79 & 89 & 99
\end{bmatrix}.$$

Example 6. ESODLS CMS number 14 881. Note this CMS corresponds to a 180 degree

rotation of ESODLS CMS number 1.

$$\begin{bmatrix} 99 & 89 & 79 & 69 & 59 & 49 & 39 & 29 & 19 & 9 \\ 98 & 88 & 78 & 68 & 58 & 48 & 38 & 28 & 18 & 8 \\ 97 & 87 & 77 & 67 & 57 & 47 & 37 & 27 & 17 & 7 \\ 96 & 86 & 76 & 66 & 56 & 46 & 36 & 26 & 16 & 6 \\ 95 & 85 & 75 & 65 & 55 & 45 & 35 & 25 & 15 & 5 \\ 94 & 84 & 74 & 64 & 54 & 44 & 34 & 24 & 14 & 4 \\ 93 & 83 & 73 & 63 & 53 & 43 & 33 & 23 & 13 & 3 \\ 92 & 82 & 72 & 62 & 52 & 42 & 32 & 22 & 12 & 2 \\ 91 & 81 & 71 & 61 & 51 & 41 & 31 & 21 & 11 & 1 \\ 90 & 80 & 70 & 60 & 50 & 40 & 30 & 20 & 10 & 0 \end{bmatrix}.$$

All 134 CNFs produced by these two CMSs were additionally parallelized by the Cube-and-Conquer (CnC) method proposed by Heule et al. [17]. To choose the best CnC parameters, we ran the algorithm by Zaikin [41] with the following parameters on the first 10 CNFs:

1. lookahead = MARCH_CU [18];
2. cdcl = CLASP-CRAFTY;
3. $k = 5$;
4. min_cubes = 10 000;
5. max_cubes = 50 000;
6. min_ref = 0;
7. $N = 1000$;
8. max_t = 600 seconds;
9. cpu_cores = 16.

As a result, on 9 CNFs the cutoff threshold of 925 turned out to be the best, while on the remaining CNF a cutoff of 930 was the best. For simplicity, we decided to use a cutoff of 925 for all 134 CNFs. Given this threshold, MARCH_CU produced about 15 000 simplified CNFs from each CNF, and then all solutions of these simplified CNFs were found by CLASP-CRAFTY on the supercomputer. For ESODLS CMS 1, the experiment took 7 days and 1 hour, while for ESODLS CMS 14 881, it took 7 days and 3 hours. Exactly 33 744 pairs of MODLS and 33 744 distinct ESODLS were found in both cases—the equivalence in both of these cases is explained by the symmetry noted in Example 6. On the other hand, these two sets of ESODLSs do not intersect, so 67 488 distinct ESODLSs were found. Based on each set, 25 009 main classes are formed. These sets of classes intersect but do not coincide since there are 30 502 main classes formed using these 67 488 ESODLSs. Recall, that finding pairs of

MODLS is the second step of algorithm B from Section 3, while collecting distinct ESODLSs from the pairs is the third step.

As for the remaining 15 358 ESODLS CMSs, all solutions of the corresponding 1 028 986 CNFs were found by CLASP-CRAFTY on a supercomputer directly without Cube-and-Conquer. This experiment took 8 days and 17 hours; on the hardest CNF the runtime was 34 hours, while on average it took 5 hours. During this experiment, 8 664 pairs of MODLS and 8 656 distinct ESODLSs were found. The corresponding 2 738 main classes produced from these 8 656 ESODLSs are new compared to 30 502 main classes mentioned above, so there are 33 240 main classes of ESODLSs of order 10.

It turned out that there are 8 ESODLSs that form two pairs of MODLS (from two different ESODLS CMSs). One of these ESODLSs and the corresponding pairs of MODLS are shown in Examples 7 and 8.

Example 7. An ESODLS of order 10 and the first pair of MODLS it forms.

$$\begin{bmatrix} 0 & 9 & 3 & 7 & 2 & 4 & 8 & 6 & 5 & 1 \\ 8 & 1 & 6 & 2 & 5 & 3 & 7 & 9 & 0 & 4 \\ 5 & 8 & 2 & 0 & 9 & 7 & 1 & 3 & 4 & 6 \\ 9 & 4 & 1 & 3 & 6 & 8 & 2 & 0 & 7 & 5 \\ 7 & 3 & 0 & 5 & 4 & 6 & 9 & 1 & 2 & 8 \\ 3 & 6 & 4 & 1 & 7 & 5 & 0 & 8 & 9 & 2 \\ 2 & 0 & 5 & 8 & 1 & 9 & 6 & 4 & 3 & 7 \\ 1 & 2 & 9 & 4 & 8 & 0 & 5 & 7 & 6 & 3 \\ 6 & 5 & 7 & 9 & 3 & 1 & 4 & 2 & 8 & 0 \\ 4 & 7 & 8 & 6 & 0 & 2 & 3 & 5 & 1 & 9 \end{bmatrix} \begin{bmatrix} 3 & 0 & 2 & 5 & 1 & 8 & 6 & 9 & 4 & 7 \\ 1 & 2 & 4 & 3 & 9 & 0 & 8 & 7 & 6 & 5 \\ 8 & 4 & 5 & 7 & 3 & 6 & 9 & 1 & 0 & 2 \\ 5 & 9 & 6 & 4 & 7 & 3 & 0 & 8 & 2 & 1 \\ 7 & 8 & 1 & 2 & 6 & 5 & 4 & 3 & 9 & 0 \\ 9 & 6 & 3 & 0 & 4 & 7 & 2 & 5 & 1 & 8 \\ 6 & 5 & 0 & 9 & 8 & 2 & 1 & 4 & 7 & 3 \\ 4 & 7 & 8 & 1 & 2 & 9 & 5 & 0 & 3 & 6 \\ 0 & 3 & 9 & 6 & 5 & 1 & 7 & 2 & 8 & 4 \\ 2 & 1 & 7 & 8 & 0 & 4 & 3 & 6 & 5 & 9 \end{bmatrix}.$$

Example 8. The ESODLS from Example 7 and the second pair of MODLS it forms.

$$\begin{bmatrix} 0 & 9 & 3 & 7 & 2 & 4 & 8 & 6 & 5 & 1 \\ 8 & 1 & 6 & 2 & 5 & 3 & 7 & 9 & 0 & 4 \\ 5 & 8 & 2 & 0 & 9 & 7 & 1 & 3 & 4 & 6 \\ 9 & 4 & 1 & 3 & 6 & 8 & 2 & 0 & 7 & 5 \\ 7 & 3 & 0 & 5 & 4 & 6 & 9 & 1 & 2 & 8 \\ 3 & 6 & 4 & 1 & 7 & 5 & 0 & 8 & 9 & 2 \\ 2 & 0 & 5 & 8 & 1 & 9 & 6 & 4 & 3 & 7 \\ 1 & 2 & 9 & 4 & 8 & 0 & 5 & 7 & 6 & 3 \\ 6 & 5 & 7 & 9 & 3 & 1 & 4 & 2 & 8 & 0 \\ 4 & 7 & 8 & 6 & 0 & 2 & 3 & 5 & 1 & 9 \end{bmatrix} \begin{bmatrix} 2 & 1 & 3 & 4 & 0 & 9 & 7 & 8 & 5 & 6 \\ 0 & 3 & 5 & 2 & 8 & 1 & 9 & 6 & 7 & 4 \\ 9 & 5 & 4 & 6 & 3 & 7 & 8 & 0 & 1 & 2 \\ 4 & 8 & 7 & 5 & 6 & 3 & 1 & 9 & 2 & 0 \\ 6 & 9 & 0 & 3 & 7 & 4 & 5 & 2 & 8 & 1 \\ 8 & 7 & 2 & 1 & 5 & 6 & 3 & 4 & 0 & 9 \\ 7 & 4 & 1 & 8 & 9 & 2 & 0 & 5 & 6 & 3 \\ 5 & 6 & 9 & 0 & 2 & 8 & 4 & 1 & 3 & 7 \\ 1 & 2 & 8 & 7 & 4 & 0 & 6 & 3 & 9 & 5 \\ 3 & 0 & 6 & 9 & 1 & 5 & 2 & 7 & 4 & 8 \end{bmatrix}.$$

As a result of these three experiments, 76 152 pairs of MODLS and 76 144 distinct ESODLSs of order 10 were found. The fourth step of algorithm B gives the number of ESODLSs of order 10 with the first row in ascending order (see the first sequence at the beginning of this section).

Since there are half a billion ESODLSs of order 10 with the first row in ascending order we needed to store them efficiently. There are 100 integers in an ESODLS of order 10. In our

C++ implementation, we divided them into 25 groups of 4 numbers each. Each group was stored as a 2 byte variable of the type `short int`. As a result, we needed about 24 gigabytes to store half a billion ESODLSs.

5.3 On a triple of MODLS of order 10 containing an ESODLS

Consider all 510 566 400 ESODLSs of order 10 with the first row in ascending order. For each such ESODLS, we found all its diagonal orthogonal mates by the Euler–Parker approach [32]. According to this approach, all transversals of a given Latin square of order n are constructed and then subsets of n disjoint transversals are found as an exact cover problem. To solve these exact cover problems, we applied Algorithm X and the dancing links technique proposed by Knuth [22]. We used a parallel version of the implementation by Kochemazov et al. [23] for this purpose. Then for each ESODLS A with at least two orthogonal mates, all possible triples of diagonal Latin squares (A, B, C) were formed where B and C are orthogonal mates of A. In none of these triples B and C were orthogonal, thereby proving the nonexistence of a triple of MODLS of order 10 containing an ESODLS. This experiment took 3 hours 30 minutes on a computer equipped with two 96-core AMD EPYC 9654 CPUs.

Zaikin et al. [43] found a triple of diagonal Latin squares (A, B, C) of order 10, such that A is orthogonal to B and C, and B is "close" to being orthogonal to C, with 74 different ordered pairs of elements out of 100. In the aforementioned triples with at least one ESODLS, there are at most 16 different ordered pairs of elements for B and C, so the triple by Zaikin et al. [43] is still the closest to a triple of MODLS of order 10 found so far.

6 Related studies

Egan and Wanless [12] enumerated MOLS of order 9. Lam et al. [25] proved the nonexistence of nine MOLS of order 10, and later Bright et al. [4] verified their result using a SAT approach. Graham and Roberts [15] enumerated SOLSs of order up to 9, and Burger et al. [9] enumerated SOLSs of order 10. Vatutin and Belyshev [36] enumerated ESODLSs of order up to 8. McKay et al. [29], Gill and Wanless [14], and Wanless [40] proved that a triple of MOLS of order 10 does not exist for certain types of Latin squares or pairs of MOLS.

Recently, various combinatorial designs based on Latin squares have been studied using SAT solvers. Liu et al. [26] investigated the existence of Holey Latin squares of order up to 14, while Lu et al. [27] and Jin et al. [20] investigated the existence of doubly SOLSs and Costas Latin squares of order 10, respectively. Huang et al. [19] applied SAT and constraint programming to prove the non-existence of some orthogonal golf designs, which are closely connected with Latin squares. Appa et al. [1] found pairs and triples of MOLS of small order by constraint programming and integer programming. Bright, Keita, and Stevens [6] used a SAT encoding of orthogonality based on Mann's product theorem (see Theorem 4) in an attempt to rule out the existence of a triple of MOLS of order 10 in which one square contains a Latin subsquare of order 4. Bright, Keita, and Stevens [7] also used the same encoding to

enumerate all pairs of MOLS of order 10 whose corresponding incidence matrices have two nontrivial relations.

7 Conclusions

In this paper we propose an enumeration algorithm aimed at ESODLSs. After applying symmetry breaking, we reduce the most time-consuming part of the algorithm to a call to an AllSAT solver. Using our algorithm, we enumerate ESODLSs of order up to 10, thus extending three existing sequences in the OEIS. In addition, we show that a triple of MODLS of order 10 containing an ESODLS does not exist.

8 Acknowledgments

Oleg Zaikin was funded by the Ministry of Science and Higher Education of the Russian Federation, project No. 121041300065-9. Curtis Bright was supported by an NSERC Discovery Grant. The authors thank the anonymous reviewer for valuable and thorough comments.

References

[1] G. Appa, D. Magos, and I. Mourtos, Searching for mutually orthogonal Latin squares via integer and constraint programming, *European J. Oper. Res.* **173** (2006), 519–530.

[2] F. E. Bennett, B. Du, and H. Zhang, Existence of self-orthogonal diagonal Latin squares with a missing subsquare, *Discrete Math.* **261** (2003), 69–86.

[3] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, IOS Press, second edition, 2021.

[4] C. Bright, K. K. H. Cheung, B. Stevens, I. S. Kotsireas, and V. Ganesh, A SAT-based resolution of Lam's problem, In *Proc. AAAI 2021*, pp. 3669–3676. AAAI Press, 2021.

[5] C. Bright, J. Gerhard, I. S. Kotsireas, and V. Ganesh, Effective problem solving using SAT solvers, In *Proc. Maple in Mathematics Education and Research 2019*, Vol. 1125 of *Commun. Comput. Inf. Sci.*, pp. 205–219. Springer, 2019.

[6] C. Bright, A. Keita, and B. Stevens, Myrvold's results on orthogonal triples of 10×10 Latin squares: A SAT investigation, arXiv preprint arXiv:2503.10504 [math.CO], 2025. Available at https://arxiv.org/abs/2503.10504.

[7] C. Bright, A. Keita, and B. Stevens, Orthogonal Latin squares of order 10 with two relations: A SAT investigation, *Discrete Math. Algorithms Appl.* (2025), to appear.

[8] J. W. Brown, F. Cherry, L. Most, E. T. Parker, and W. D. Wallis, Completion of the spectrum of orthogonal diagonal Latin squares. In *Graphs, Matrices, and Designs*, pp. 43–50. Routledge, 1993.

[9] A. P. Burger, M. P. Kidd, and J. H. van Vuuren, Enumerasie van self-ortogonale Latynse vierkante van orde 10, *Litnet Akademies (Natuurwetenskappe)* **1** (2010), 1–22.

[10] S. A. Cook, The complexity of theorem-proving procedures, In *Proc. STOC 1971*, pp. 151–158. ACM, 1971.

[11] W. F. Dowling and J. H. Gallier, Linear-time algorithms for testing the satisfiability of propositional Horn formulae, *J. Log. Program.* **1** (1984), 267–284.

[12] J. Egan and I. M. Wanless, Enumeration of MOLS of small order, *Math. Comput.* **85** (2016), 799–824.

[13] M. Gebser, B. Kaufmann, and T. Schaub, Conflict-driven answer set solving: From theory to practice, *Artificial Intelligence* **187** (2012), 52–89.

[14] M. J. Gill and I. M. Wanless, Pairs of MOLS of order ten satisfying non-trivial relations, *Des. Codes Cryptogr.* **91** (2023), 1293–1313.

[15] G. P. Graham and C. E. Roberts, Enumeration and isomorphic classification of self-orthogonal Latin squares, *J. Combin. Math. Combin. Comput.* **59** (2010), 101–118.

[16] A. Hedayat, An application of sum composition: A self orthogonal Latin Square of order ten, *J. Comb. Theory A* **14** (1973), 256–260.

[17] M. Heule, O. Kullmann, and A. Biere, Cube-and-Conquer for Satisfiability. In *Handbook of Parallel Constraint Reasoning*, pp. 31–59. Springer, 2018.

[18] M. Heule and H. van Maaren, March_dl: Adding adaptive heuristics and a new branching strategy, *Journal on Satisfiability, Boolean Modeling and Computation* **2** (2006), 47–59.

[19] P. Huang, M. Liu, C. Ge, F. Ma, and J. Zhang, Investigating the existence of orthogonal golf designs via Satisfiability testing, In *Proc. ISSAC*, pp. 203–210. ACM, 2019.

[20] J. Jin, Y. Lv, C. Ge, F. Ma, and J. Zhang, Investigating the existence of Costas Latin squares via satisfiability testing, In *Proc. SAT 2021*, Vol. 12831 of *Lect. Notes in Comp. Sci.*, pp. 270–279. Springer, 2021.

[21] A. D. Keedwell and J. Dénes, *Latin Squares and their Applications*, North-Holland, second edition, 2015.

[22] D. Knuth, Dancing links. In *Millennial Perspectives in Computer Science*, pp. 187–214. Red Globe Press, 2000.

[23] S. Kochemazov, O. Zaikin, and A. Semenov, Runtime estimation for enumerating all mutually orthogonal diagonal Latin squares of order 10, In *Proc. MIPRO 2017*, pp. 1166–1171. IEEE, 2017.

[24] S. Kochemazov, O. Zaikin, E. Vatutin, and A. Belyshev, Enumerating diagonal Latin squares of order up to 9, *J. Integer Seq.* **23** (2020), 20.1.2.

[25] C. W. H. Lam, L. Thiel, and S. Swierz, The nonexistence of finite projective planes of order 10, *Canad. J. Math.* **41** (1989), 1117–1123.

[26] M. Liu, R. Han, F. Jia, P. Huang, F. Ma, H. Zhang, and J. Zhang, Investigating the existence of Holey Latin squares via satisfiability testing, In *Proc. PRICAI 2023*, Vol. 14326 of *Lect. Notes in Comp. Sci.*, pp. 410–422. Springer, 2023.

[27] R. Lu, S. Liu, and J. Zhang, Searching for doubly self-orthogonal Latin squares, In *Proc. CP 2011*, Vol. 6876 of *Lect. Notes in Comp. Sci.*, pp. 538–545. Springer, 2011.

[28] H. B. Mann, The construction of orthogonal Latin squares, *Ann. Math. Stat.* **13** (1942), 418–423.

[29] B. D. McKay, A. Meynert, and W. Myrvold, Small Latin squares, quasigroups, and loops, *J. Combin. Des.* **15** (2007), 98–119.

[30] V.-H. Nguyen, V.-Q. Nguyen, K. Kim, and P. Barahona, Empirical study on SAT-encodings of the at-most-one constraint, In *Proc. SMA 2020*, pp. 470–475. ACM, 2020.

[31] A. Novopashin, Irkutsk supercomputer center of SB RAS, 2025. Available at https://hpc.icc.ru.

[32] E. T. Parker, Computer investigation of orthogonal Latin squares of order ten, *Proc. Symp. Appl. Math.* **15** (1963), 73–81.

[33] N. Rubin, C. Bright, K. K. H. Cheung, and B. Stevens, Improving integer and constraint programming for Graeco-Latin squares, In *Proc. ICTAI 2021*, pp. 604–608. IEEE, 2021.

[34] N. J. A. Sloane, The on-line encyclopedia of integer sequences, 2025. Available at https://oeis.org.

[35] M. Soos, K. Nohl, and C. Castelluccia, Extending SAT solvers to cryptographic problems, In *Proc. SAT 2009*, Vol. 5584 of *Lect. Notes in Comp. Sci.*, pp. 244–257. Springer, 2009.

[36] E. Vatutin and A. Belyshev, Enumerating the orthogonal diagonal Latin squares of small order for different types of orthogonality, In *Proc. RuSCDays 2020*, Vol. 1331 of *Commun. Comput. Inf. Sci.*, pp. 586–597. Springer, 2020.

[37] E. Vatutin and O. Zaikin, Classification of cells mapping schemes related to orthogonal diagonal Latin squares of small order, In *Proc. RuSCDays 2023*, Vol. 14389 of *Lect. Notes in Comp. Sci.*, pp. 21–34. Springer, 2023.

[38] E. Vatutin, O. Zaikin, S. Kochemazov, and S. Valyaev, Using volunteer computing to study some features of diagonal Latin squares, *Open Engineering* **7** (2017), 453–460.

[39] E. Vatutin, O. Zaikin, M. Manzyuk, and N. Nikitina, Searching for orthogonal Latin squares via cells mapping and BOINC-based Cube-and-Conquer, In *Proc. RuSCDays 2021*, Vol. 1510 of *Commun. Comput. Inf. Sci.*, pp. 498–512. Springer, 2021.

[40] I. M. Wanless, Answers to questions by Dénes on Latin power sets, *Eur. J. Comb.* **22** (2001), 1009–1020.

[41] O. Zaikin, Inverting 43-step MD4 via cube-and-conquer, In *Proc. IJCAI 2022*, pp. 1894–1900. ijcai.org, 2022.

[42] O. Zaikin, Searching for combinatorial designs based on Latin squares, 2025. Available at https://github.com/olegzaikin/latinsquares.

[43] O. Zaikin, A. Zhuravlev, S. Kochemazov, and E. Vatutin, On the construction of triples of diagonal Latin squares of order 10, *Electronic Notes in Discrete Mathematics* **54** (2016), 307–312.

2020 *Mathematics Subject Classification*: Primary 05B15; Secondary 68T27.
Keywords: Latin square, orthogonality, enumeration, symmetry breaking, SAT, AllSAT.

(Concerned with sequences A309598, A309599, A309210, A299784, and A309283.)