

Searching for Orthogonal Latin Squares via Cells Mapping and BOINC-based Cube-and-Conquer

Eduard Vatutin¹[0000-0002-7362-7387], Oleg Zaikin²[0000-0002-0145-5010]✉, Maxim Manzyuk³[0000-0002-6628-0119], and Natalia Nikitina⁴[0000-0002-0538-2939]

¹ Southwest State University, Kursk, Russia
evatutin@rambler.ru

² Swansea University, Swansea, United Kingdom
o.s.zaikin@swansea.ac.uk

³ Internet portal BOINC.ru, Moscow, Russia
hoarfrost@rambler.ru

⁴ Institute of Applied Mathematical Research, Karelian Research Centre of RAS,
Petrozavodsk, Russia
nikitina@krc.karelia.ru

Abstract. This study focuses on searching for pairs of orthogonal diagonal Latin squares of order 10. Consider a cells mapping in accordance to which one diagonal Latin square is mapped to another one. Given a certain cells mapping schema, the problem is to find a pair of orthogonal diagonal Latin squares of order 10 such that they match the schema (or to prove that such a pair does not exist). The problem is reduced to the Boolean satisfiability problem (SAT). Three mapping schemes are considered, and for each of them a SAT instance is constructed. If a satisfying assignment is found for an instance, the corresponding pair of orthogonal Latin squares can be easily extracted from it. The Cube-and-Conquer approach is used to solve the instances. The cubing phase is performed on a sequential look-ahead SAT solver, while on the conquer phase an experiment in a BOINC-based volunteer computing project is launched. In the experiment, for two out of three schemes orthogonal pairs are found.

Keywords: Volunteer computing · BOINC · Latin square · MOLS · SAT · Cube-and-Conquer

1 Introduction

Volunteer computing [1] is a type of distributed computing that uses resources owned by private persons. Alternatively, volunteer computing can be considered as a type of desktop grid computing [2]. Volunteer computing is a quite cheap and natural approach to solving computationally hard problems that can be decomposed into independent subproblems. Such problems appear, for example, in astronomy, medicine, cryptography, and combinatorics. During the last two decades, a number of important and challenging problems from these areas were successfully solved in volunteer computing projects. Most of such projects are

based on BOINC (Berkeley Open Infrastructure for Network Computing [3]). Note, that BOINC is quite flexible and can be also used to create enterprise desktop grids (see e.g. [4]). Though launching and maintaining a BOINC-based volunteer computing project is not an easy task, if its team cope with that and also take into account volunteers' feedback, the project can attract significant computational resources [5].

Latin square is a very simple yet important combinatorial design [6]. One of the most well-studied property between two Latin squares is orthogonality. Many theoretical and practical problems (e.g. from cryptography and coding theory) can be reduced to finding systems of orthogonal Latin squares. One of the most well-known open mathematical problem is to determine the existence of a triple of mutually orthogonal Latin squares of order 10.

Though as a rule "pure" combinatorial algorithms are applied for finding systems of orthogonal Latin squares (see e.g. [7,8]), it is also possible to reduce these problems to the Boolean satisfiability problem (SAT [9]). Despite significant recent progress in complete SAT solving algorithms [10,11], even the best SAT solvers can (sometimes) show competitive results only if there are few to no solutions in the considered problem related to orthogonal Latin squares. It means that the corresponding SAT instance has only few satisfying assignments or it is unsatisfiable, respectively. Note that even in this case such SAT instances in practice are very hard, that is why distributed algorithms are crucial to deal with them. Recently, Cube-and-Conquer has showed itself as a very powerful distributed SAT solving algorithm [12]. In this study, a BOINC-based Cube-and-Conquer SAT solver is implemented and applied to finding orthogonal diagonal Latin squares.

A brief outline of the paper is as follows. In Section 2, preliminary information regarding orthogonal diagonal Latin squares is given. Section 3 describes how cells mapping can be used to find orthogonal (diagonal) Latin squares. In Section 4 the cells mapping approach is reduced to SAT, and several cells mapping schemes are investigated via the BOINC-based Cube-and-Conquer SAT solver. Finally, related works are discussed and conclusions are drawn.

2 Orthogonal Diagonal Latin Squares

A *Latin square* of order N is a square table $N \times N$ filled with N different symbols (e.g. $0, \dots, N - 1$) in such a way, that all symbols within a single row or single column are distinct [6]. A *diagonal Latin square* is a Latin square in which all symbols in both main diagonal and anti-diagonal are distinct. A *transversal* of a Latin square of order N is a set of N entries, one selected from each row and each column such that no two entries contain the same symbol.

All Latin squares can be divided into *isotopy classes* such that any Latin square from an isotopy class can be produced from any other Latin square from the same class by permuting rows, columns, or symbols names. *Isotopism* is an equivalence relation. Another equivalence relation on the set of Latin squares is *main class isotopism*. The corresponding classes are called *main classes*. Each

such class contains up to six isotopy classes. Latin square that is a lexicographically minimal representative of a main class is called a *canonical form*.

Two Latin squares $A = (a_{ij}), B = (b_{ij})$ of order N are *orthogonal* if all ordered pairs $(a_{ij}, b_{ij}), 0 \leq i, j \leq N-1$ are distinct. A set of Latin squares of the same order, all pairs of which are orthogonal, is called a set of *mutually orthogonal Latin squares* (MOLS). For diagonal Latin squares, *MODLS* is defined similarly. MODLS are quite rare compared to MOLS. The first pair of MODLS of order 10 was presented in [13].

As a rule, searching for an orthogonal mate for a given Latin square A of order N is performed via the Euler-Parker method (see e.g. [14]). According to this method, the orthogonal mate can be easily constructed from a set of N disjoint transversals of A , so the goal is to find such a set. It should be noted, that searching for disjoint transversals is the most time-consuming part of the Euler-Parker method. In [15] the problem of searching for a set of N disjoint diagonal transversals (required for finding MODLS) was reduced to the exact cover problem that in turn was solved by DLX, an implementation of the Algorithm X [16].

Self-orthogonal Latin square (SOLS) denotes a Latin square that is orthogonal to its transpose (see e.g. [17]). It gives another approach for finding MOLS. For diagonal Latin squares, SODLS is defined similarly. *Enhanced self-orthogonal diagonal Latin square* (ESODLS) denotes a diagonal Latin square that is orthogonal to some diagonal Latin square from the same main class [18]. It is clear that ESODLS is a generalisation of SODLS and can be also used to find MODLS without dealing with transversals.

It is also possible to reduce finding MOLS or MODLS to Integer Programming [19,20], Constraint Programming [19,20], or SAT [9,21]. The resulted instances are solved via solvers from the corresponding areas, and then the solution of the original problem is constructed. In the present paper, the SAT approach is employed for finding new pairs of MODLS of order 10.

3 Finding Orthogonal Latin Squares via Cells Mapping Schemes

In this section, cells mapping schemes for Latin squares are proposed, then such schemes for ESODLS are described, and finally a search for ESODLS cells mapping schemes of order 10 is discussed.

3.1 Cells Mapping Schema

Consider two Latin squares A and B of order N . Assume that in both squares cells are numerated from 0 to $N^2 - 1$ (left-to-right, top-to-bottom), and $A[i]$ denotes value of i -th cell, $0 \leq i \leq N^2 - 1$ (similarly for B). A *cells mapping schema* (CMS) for an ordered pair of squares (A, B) is a permutation p of N^2 integer numbers $0, \dots, N^2 - 1$ such that $p[i] = j, 0 \leq i, j \leq N^2 - 1$ iff $A[i] = B[j]$. Therefore p shows how cells of A can be mapped onto cells of B . This fact

is denoted as $p(A) = B$. Note, that any CMS for order N can be naturally represented not only as a permutation of size N^2 , but also as a $N \times N$ table. A pair of Latin squares A and B *matches* a CMS (or, interchangeably, CMS *matches* the pair of Latin squares) if A can be mapped onto B via the CMS.

Example 1. CMS

$$(0, 5, 10, 15, 4, 1, 14, 11, 8, 13, 2, 7, 12, 9, 6, 3).$$

is equal to

$$\begin{pmatrix} 0 & 5 & 10 & 15 \\ 4 & 1 & 14 & 11 \\ 8 & 13 & 2 & 7 \\ 12 & 9 & 6 & 3 \end{pmatrix}.$$

The following pair of Latin squares matches the CMS:

$$A = \begin{pmatrix} 0 & 2 & 3 & 1 \\ 3 & 1 & 0 & 2 \\ 1 & 3 & 2 & 0 \\ 2 & 0 & 1 & 3 \end{pmatrix}, B = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \end{pmatrix}$$

It should be noted that for some CMS enormous number of pairs of Latin squares can be matched, while for others no such pairs can be matched at all. In the above example two Latin squares are orthogonal. An example of the CMS with no matched MOLS is a *trivial* CMS, i.e. such a CMS p that $p[i] = i, 0 \leq i \leq N^2 - 1$. For $N > 1$, a trivial CMS does not give an orthogonal mate since it maps a Latin square to itself (if $N = 1$, the single existing Latin square is orthogonal to itself).

It is clear that there are $(N^2)!$ different CMS for order N . A *fixed point* in a CMS p is such a number i that $p[i] = i$ (see e.g. [22]). Note, that for order N a trivial CMS has N^2 fixed points. It can be shown that if a CMS matches some pair of MOLS, then the CMS has at most N fixed points. It follows from the fact that for order N there are only N different variants of fixed points: $(0, 0), (1, 1), \dots, (N - 1, N - 1)$. Therefore if there are more than N fixed points, then at least one of these pairs of symbols will occur more than once, thus violating the orthogonality condition.

Given an arbitrary pair of MOLS A and B , a CMS p can be extracted from it. For this purpose it is required to find such cells with numbers i_1 and i_2 , for which $A[i_1] = v_1, B[i_1] = v_2, A[i_2] = v_2, B[i_2] = v_1, p[i_1] = i_2, p[i_2] = i_1$. Such CMS are the simplest ones that match some MOLS; in each such CMS there are N 1-cycles and $N^2 - N$ 2-cycles. These CMS are called *canonical*. It can be shown that if two Latin squares A and B are orthogonal, and a canonical CMS maps A onto B , then the CMS also maps B onto A . This fact is denoted by $p(A) = B, p(B) = A$.

3.2 ESODLS CMS

CMS can be naturally connected with SODLS (see Section 2).

Example 2. Consider a SODLS of order 4 with cells numerated as follows:

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{pmatrix}.$$

The corresponding CMS in the form of a 4×4 table is

$$\begin{pmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{pmatrix}$$

The same CMS in the form of a permutation is

$$(0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15).$$

CMS can also be connected with ESODLS (see Section 2). By applying to a trivial CMS all possible equivalent transformations, which are used to construct a main class of diagonal Latin squares given any its representative [23], a set of CMS is obtained that is called a set of *ESODLS CMS*. The number of ESODLS CMS for order N is equal to the maximal size of main classes of diagonal Latin squares of order N with the fixed first row, see sequence A299784⁵ in the online encyclopedia of integer sequences (OEIS) [24]. For order 10, there are 15 360 ESODLS CMS.

Orthogonality cycle of length k is such a set of Latin squares $\{L_1, \dots, L_k\}$ that L_i is orthogonal to L_{i+1} for $1 \leq i \leq k - 1$, and finally L_k is orthogonal to L_1 . It is clear that a canonical CMS gives an orthogonal cycle of length 2. At the present moment, only orthogonal cycles of length 2 and 4 are known for diagonal Latin squares of order 10 [25].

Some ESODLS CMS are not canonical, but they also can have matching pairs of MODLS. For example, during the search for MODLS of order 10 in neighbourhoods of generalised symmetries, 2 orthogonal cycles of length 4 were found [26]. In each of these orthogonal cycles diagonal Latin squares are obtained from each other by applying one of two ESODLS CMS: one direction for one CMS. Figure 1 shows these orthogonal cycles. Each ESODLS CMS from the figure has 25 4-cycles, so they are not canonical. It can be shown that a consequent applying of a CMS forms an orthogonal cycle of length equal to least common multiple of cycles lengths in the CMS. Theoretically, for order 10 from some ESODLS CMS (e.g. with 10 1-cycles and 30 3-cycles) it would be possible to obtain an orthogonal cycle of length 3, i.e. a triple of MODLS of order 10.

⁵ <https://oeis.org/A299784>

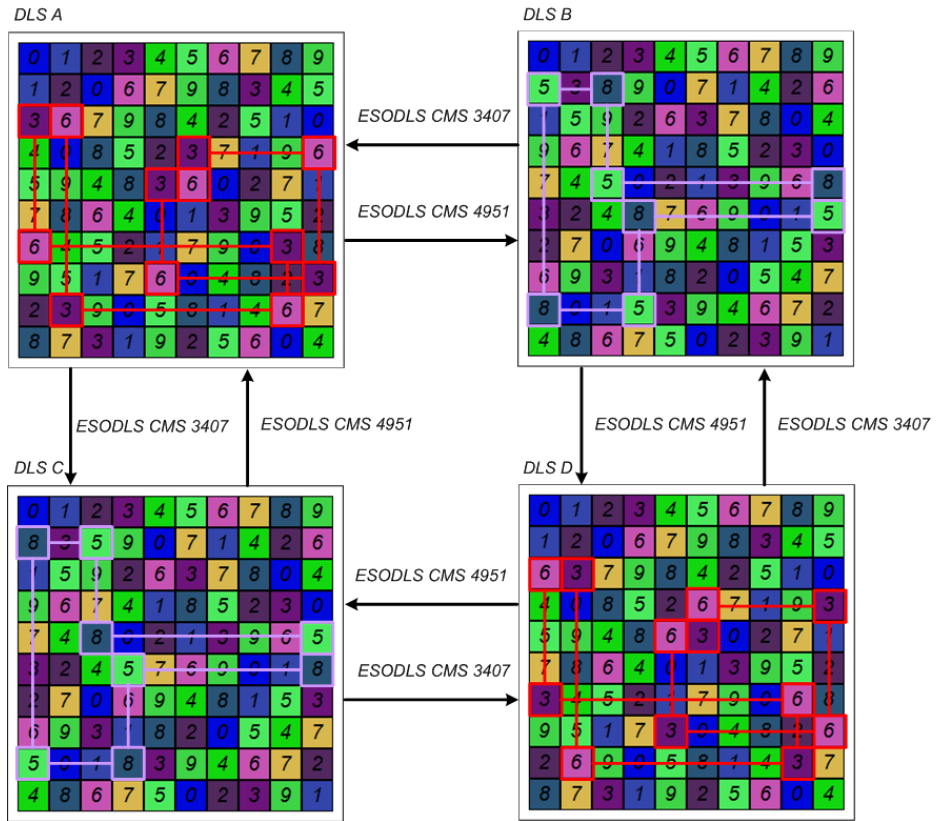


Fig. 1. An example of an orthogonal cycle of four diagonal Latin squares (DLS) and two CMS. DLS stands for diagonal Latin square.

Two CMS from the figure are as follows:

$$CMS_{3407} = \begin{pmatrix} 9 & 69 & 19 & 59 & 29 & 79 & 49 & 89 & 39 & 99 \\ 3 & 63 & 13 & 53 & 23 & 73 & 43 & 83 & 33 & 93 \\ 8 & 68 & 18 & 58 & 28 & 78 & 48 & 88 & 38 & 98 \\ 4 & 64 & 14 & 54 & 24 & 74 & 44 & 84 & 34 & 94 \\ 7 & 67 & 17 & 57 & 27 & 77 & 47 & 87 & 37 & 97 \\ 2 & 62 & 12 & 52 & 22 & 72 & 42 & 82 & 32 & 92 \\ 5 & 65 & 15 & 55 & 25 & 75 & 45 & 85 & 35 & 95 \\ 1 & 61 & 11 & 51 & 21 & 71 & 41 & 81 & 31 & 91 \\ 6 & 66 & 16 & 56 & 26 & 76 & 46 & 86 & 36 & 96 \\ 0 & 60 & 10 & 50 & 20 & 70 & 40 & 80 & 30 & 90 \end{pmatrix},$$

$$CMS_{4951} = CMS_{3407}^{-1} = \begin{pmatrix} 90 & 70 & 50 & 10 & 30 & 60 & 80 & 40 & 20 & 0 \\ 92 & 72 & 52 & 12 & 32 & 62 & 82 & 42 & 22 & 2 \\ 94 & 74 & 54 & 14 & 34 & 64 & 84 & 44 & 24 & 4 \\ 98 & 78 & 58 & 18 & 38 & 68 & 88 & 48 & 28 & 8 \\ 96 & 76 & 56 & 16 & 36 & 66 & 86 & 46 & 26 & 6 \\ 93 & 73 & 53 & 13 & 33 & 63 & 83 & 43 & 23 & 3 \\ 91 & 71 & 51 & 11 & 31 & 61 & 81 & 41 & 21 & 1 \\ 95 & 75 & 55 & 15 & 35 & 65 & 85 & 45 & 25 & 5 \\ 97 & 77 & 57 & 17 & 37 & 67 & 87 & 47 & 27 & 7 \\ 99 & 79 & 59 & 19 & 39 & 69 & 89 & 49 & 29 & 9 \end{pmatrix}.$$

For (non-diagonal) Latin squares ESOLS CMS can be constructed similarly, but their number will likely be much higher than that for diagonal Latin squares. In the volunteer computing project RakeSearch [27], pairs of MODLS were found based on permutation of rows. In fact these pairs are ESOLS by not ESODLS, because the corresponding canonical forms of Latin squares coincide, but the canonical forms of diagonal Latin squares can differ.

ESODLS CMS are close to generalised symmetries/automorphisms of diagonal Latin squares. Each of them gives some transformation F that maps cells $(i, j) \rightarrow (i', j')$. In case of generalised symmetries, cells of a diagonal Latin square A with symmetry are mapped onto themselves: $f(A) = A$ (case 1), while orthogonal diagonal Latin squares are found by the Euler-Parker method. For ESODLS CMS $f(A) = B, f(B) = A$ (case 2), where A and B are orthogonal diagonal Latin squares from the same main class.

3.3 Search for ESODLS CMS of Order 10

Given a CMS, it is possible to construct all matching pairs of MOLS by the depth-first search. Note, that transversals are not required in this case. For some CMS all matching pairs of MOLS can be found in few seconds, yet for other ones the search can take days. The runtime can be significantly reduced via the usage of a dedicated implementation with bit arithmetic and nested loops (see e.g. [28]). One more further step is a GPU-implementation. For ESODLS CMS the runtime (and the number of matching orthogonal pairs) is usually several orders of magnitude higher than that for an arbitrary CMS. Note that a CMS can be obtained randomly or from known pairs of MOLS.

For about four thousand ESODLS CMS (out of 15 360) of order 10, all matching pairs of MODLS of order 10 can be found quite fast on a computer via the depth-first search. To find all pairs of MODLS for other ESODLS CMS, in the volunteer computing project Gerasim@Home [29] a computational experiment was launched. For each CMS, workunits (computational tasks in a volunteer computing project) were formed as follows: values of two cells of the first diagonal Latin square were chosen randomly such that they did not violate the constraints. As a result, values of two corresponding cells (set by the CMS) of the second diagonal Latin square were assigned automatically. Then on a volunteer

computer values of other CMS cells (98 out of 100) were filled via the depth-first search algorithm mentioned above. It turned out, that some workunits can be processed very fast, but for others the runtime can be extremely high. That is why a time limit of 10 hours was set for workunits. When the time limit was reached, all pairs of MODLS found so far (if any) were saved and returned to the server. It turned out, that for none of the studied CMS all workunits could be processed within the time limit. In the future it might be possible to reduce the search spaces without decreasing the number of solutions. As a result of the experiment, 33 240 canonical forms of ESODLS of order 10 were found. This allowed setting a new lower bound $a(10) \geq 33\,240$ in OEIS sequence A309210⁶.

4 Finding MODLS via CMS, Cube-and-Conquer, and BOINC

This section studies three ESODLS CMS of order 10, for which no pairs of MODLS were found in the experiment described in Subsection 3.3. In particular, CMS_{3407} and CMS_{4951} from Subsection 3.2 were taken, and also the following CMS:

$$CMS_{1234} = \begin{pmatrix} 27 & 26 & 21 & 25 & 29 & 20 & 24 & 28 & 23 & 22 \\ 37 & 36 & 31 & 35 & 39 & 30 & 34 & 38 & 33 & 32 \\ 87 & 86 & 81 & 85 & 89 & 80 & 84 & 88 & 83 & 82 \\ 47 & 46 & 41 & 45 & 49 & 40 & 44 & 48 & 43 & 42 \\ 7 & 6 & 1 & 5 & 9 & 0 & 4 & 8 & 3 & 2 \\ 97 & 96 & 91 & 95 & 99 & 90 & 94 & 98 & 93 & 92 \\ 57 & 56 & 51 & 55 & 59 & 50 & 54 & 58 & 53 & 52 \\ 17 & 16 & 11 & 15 & 19 & 10 & 14 & 18 & 13 & 12 \\ 67 & 66 & 61 & 65 & 69 & 60 & 64 & 68 & 63 & 62 \\ 77 & 76 & 71 & 75 & 79 & 70 & 74 & 78 & 73 & 72 \end{pmatrix}.$$

The problems of finding pairs of MODLS of order 10 that match these ESODLS CMS were reduced to SAT. In its decision version SAT is formulated as follows: for an arbitrary Boolean formula to determine if it is satisfiable or not [9]. SAT is historically the first NP-complete problem, thus it is possible to effectively reduce plenty of known problems to it. In the last three decades, SAT solvers have become very effective, that is why nowadays both practical and theoretical problems from various areas are reduced to SAT and solved by SAT solvers. In practice a Boolean formula is usually represented in Conjunctive Normal Form (CNF) that is a conjunction of disjunctions (clauses). If a SAT instance is way too hard for a sequential solver, then parallel (multithreaded) or distributed solvers can be applied to it [30].

In the remaining of the section, a SAT encoding for the considered problems is discussed, then the experimental results obtained via a Cube-and-Conquer SAT solver in a volunteer computing project are presented.

⁶ <https://oeis.org/A309210>

4.1 SAT Encoding

Consider the following problem: given a CMS of order N to find a pair of MODLS of order N that matches this CMS, or to prove that no such pair exists. First, the SAT encoding for finding a pair of MODLS from [31] was taken for this purpose. Briefly, the encoding is as follows: each of two diagonal Latin square of order N is represented as an $N \times N \times N$ incidence cube where dimensions are identified with the rows, columns, and the symbols. A cell with coordinates $(i, j, k), 0 \leq i, j, k \leq N - 1$ contains 1 iff the cell (i, j) of the Latin square contains k , 0 otherwise. It is clear that if any two coordinates in the incidence cube are fixed, then the remaining “line” contains exactly one 1. Every cube cell is naturally encoded by one Boolean variable, so each diagonal Latin square of order N is encoded by N^3 Boolean variables. A similar approach can be found e.g. in [32].

Two groups of clauses are added to the CNF: the first one reflects the diagonal Latin square conditions, while the second one reflects the orthogonality conditions. Clauses from the first group contain only variables of the corresponding square, while clauses of the second group contain variables from both squares. The so-called *naive* encoding of the orthogonality condition was used, for details see [31]. For order 10 the corresponding CNF consists of 2 000 variables and 434 440 clauses. Recall that this CNF encodes the problem of finding a pair of MODLS of order 10.

The described base CNF can be quite naturally used for constructing a CNF that encodes finding pairs of MODLS of order 10, which match a given CMS. Assume that due to CMS (see Subsection 3.1) some cell of the first square must be equal to a certain cell of the second square. Recall that each square cell is encoded by N Boolean variables, so it is needed to encode the fact that two corresponding Boolean arrays of size N are equal. Since the equality of two Boolean variables x, y is encoded by two clauses $(x \vee \neg y) \wedge (\neg x \vee y)$, $2N$ clauses are needed to encode equality of two Boolean arrays of size N . Since CMS has N^2 entries, $2N^3$ clauses are required in total to reflect the CMS conditions. Taking this into account, three CNFs were constructed by adding to the base CNF 2 000 2-literal clauses, which correspond to three considered CMS. It means that each of three CNF consisted of 2 000 variables and 436 440 clauses.

It is known that the first row in the first diagonal Latin square can be fixed in ascending order $0, \dots, N - 1$ without reducing the amount of the corresponding pairs of MODLS. This was reflected by adding 10 one-literal clauses, so finally each of three CNF consisted of 2 000 variables and 436 450 clauses.

4.2 Solving in RakeSearch via Cube-and-Conquer

All three constructed CNFs turned out to be too hard for sequential SAT solvers, that is why Cube-and-Conquer [12] was applied to them. Its parallelisation strategy is a variant of Divide-and-Conquer. According to Cube-and-Conquer, on the *cubing* phase a look-ahead SAT solver [11] is launched on a CNF and splits

the problem into *cubes*. On the *conquer* phase a simplified subproblem is constructed based on each cube, then these subproblems are solved via a Conflict-driven clause learning (CDCL) SAT solver [10]. Since cubes can be processed independently, the conquer phase can be easily parallelised.

It is usually crucial to minimise a CNF before launching a look-ahead solver. All three CNFs were minimised via the CADICAL CDCL solver [33] of version 1.3.0. In particular, this solver was launched in the minimisation mode for 1 minute on a personal computer on each CNF. The minimised CNFs (which encode exactly the same problems) had the same number of variables, but much less clauses — about 152-155 thousand instead of 436 thousand.

On the first stage, the MARCH_CU look-ahead solver [34] was launched on the minimised CNFs on a computer. The splitting parameter n was equal to 580, that value was chosen in accordance with preliminary experiments. As a result, 5 506 614, 5 507 514, and 4 144 254 cubes were generated for CMS_{3407} , CMS_{4951} , and CMS_{1234} , respectively. To exclude simple cubes, the KISSAT [33] CDCL solver of version sc2020 was launched on them with the time limit of 5 seconds. This experiment was held on a supercomputer. It turned out, that 2 265 747, 2 660 949, and 2 359 952 cubes were not processed within the time limit, respectively. To process these 7 286 648 cubes, a BOINC-based volunteer computing project RakeSearch [27] was used as an umbrella project, i.e. as a project that can solve problems posted by various scientific teams, see e.g. [35]. 7 286 648 cubes were divided into 364 334 workunits (at most 20 cubes in each).

As a computing application, a modification of the MINISAT [36] CDCL solver of version 2.2 was used. This solver had already been used earlier as a base of a similar computing application [29], so it was quite easy to implement a new computing application on top of it. Four versions of the computing application were implemented: Windows x86, Windows x64, Linux x86, and Linux x64. For each cube the limit of 5000 MINISAT restarts was set, that roughly corresponds to about 3 minutes runtime on one core of a modern desktop CPU.

The experiment started on 20 July 2020 and ended on 9 September 2020. In total, 1 611 computers of 625 volunteers took part in it. Finally, all 364 334 workunits were processed successfully. Processing of almost all cubes was interrupted due to the restart limit, but on 29 of them satisfying assignments were found. As a result, 18 pairs of MODLS of order 10 were found for CMS_{3407} and 11 ones for CMS_{4951} . The first found pair for CMS_{3407} is:

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

The first pair for CMS_{4951} is:

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

As for CMS_{1234} , processing of all corresponding cubes was interrupted due to the restart limit, so at the moment it is still unclear whether any pair of MODLS matches this CMS.

It turned out that all found pairs are isomorphic to known pairs that were found earlier in the neighborhoods for generalized symmetries (see [26]). Nevertheless, the results show that Cube-and-Conquer is able to cope with the considered combinatorial problem. All found pairs, corresponding CNFs, as well as sources of the CNF generator, workunit generator, and computing application can be found online⁷.

5 Related Works

A number of distributed Cube-and-Conquer solvers are known: aimed at BOINC-based enterprise desktop grid computing [37]; aimed at cluster and cloud computing [38]. Recently, Cube-and-Conquer was used for solving various combinatorial problems, e.g. the Boolean Pythagorean triples problem [39] and Lam's problem [40].

A parallel tree search was implemented in the volunteer computing project yoyo@home [41]. In [42] the Branch-and-Bound method was implemented in an enterprise BOINC-based desktop grid.

In the volunteer computing project SAT@home [29], pairs of MODLS of order 10 were found by a Divide-and-Conquer SAT solver. In opposite to Cube-and-Conquer, in this solver cubes are generated by a black-box optimization algorithm that employs the Monte Carlo method [43]. In some experiments, SAT@home was used in combination with computing clusters [44].

The volunteer computing project Gerasim@Home [29] was used to enumerate diagonal Latin squares of order 9 [45]. At the present moment, this project searches for canonical forms of diagonal Latin squares of order 10 [25]. In the volunteer computing project RakeSearch, an ensemble of orthogonality graphs for the set of all diagonal Latin squares of order 9 was constructed [27]. This ensemble was based on pairs of MODLS of order 9, found via row permuting techniques.

⁷ https://github.com/Nauchnik/SAT-at-home/tree/master/src_boinc_satcmsdls

6 Conclusions

The present paper proposes a new approach for finding pairs of orthogonal Latin squares. It also describes the first distributed Cube-and-Conquer SAT solver aimed at volunteer computing. The solver was applied to hard SAT instances, which encode the problem of finding pairs of orthogonal diagonal Latin squares of order 10 via the proposed approach. A computational experiment in a BOINC-based volunteer computing project made it possible to find the sought pairs.

In the future we are planning to try other look-ahead SAT solvers in the cubing phase and modern CDCL SAT solvers in the conquer phase. Also we are planning to perform more detailed comparison of the SAT solver with combinatorial algorithms on the considered problems.

Acknowledgements. Authors thank all RakeSearch and Gerasim@home volunteers, whose computers took part in the experiments. Oleg Zaikin was supported by EPSRC grant EP/S015523/1.

References

1. Anderson, D.P., Fedak, G.: The computational and storage potential of volunteer computing. In: Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006), 16-19 May 2006, Singapore. pp. 73–80. IEEE Computer Society (2006)
2. Cerin, C., Fedak, G.: Desktop Grid Computing. Chapman & Hall/CRC, 1st edn. (2012)
3. Anderson, D.P.: BOINC: A platform for volunteer computing. *J. Grid Comput.* 18(1), 99–122 (2020)
4. Ivashko, E., Chernov, I., Nikitina, N.: A survey of desktop grid scheduling. *IEEE Trans. Parallel Distrib. Syst.* 29(12), 2882–2895 (2018)
5. Yakimets, V.N., Kurochkin, I.I.: Roadmap for improving volunteer distributed computing project performance. In: Voevodin, V.V., Sobolev, S. (eds.) Supercomputing - 5th Russian Supercomputing Days, RuSCDays 2019, Moscow, Russia, September 23-24, 2019, Revised Selected Papers. Communications in Computer and Information Science, vol. 1129, pp. 690–700. Springer (2019)
6. Colbourn, C., Dinitz, J., Wanless, I., Bennett, F., Lindner, C., Julian R. Abel, R., Finizio, N., Zhu, L., Greig, M.: Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications), chap. Latin Squares, pp. 224–265. Chapman and Hall/CRC (2006)
7. McKay, B.D., Meynert, A., Myrvold, W.: Small Latin squares, quasigroups, and loops. *Journal of Combinatorial Designs* 15(2), 98–119 (2007)
8. Egan, J., Wanless, I.M.: Enumeration of MOLS of small order. *Math. Comput.* 85(298), 799–824 (2016)
9. Zhang, H.: Combinatorial designs by SAT solvers. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 533–568. IOS Press (2009)

10. Marques-Silva, J.P., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) *Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications*, vol. 185, pp. 131–153. IOS Press (2009)
11. Heule, M., van Maaren, H.: Look-ahead based SAT solvers. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) *Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications*, vol. 185, pp. 155–184. IOS Press (2009)
12. Heule, M.J.H., Kullmann, O., Biere, A.: Cube-and-Conquer for Satisfiability. In: Hamadi, Y., Sais, L. (eds.) *Handbook of Parallel Constraint Reasoning*, pp. 31–59. Springer (2018)
13. Brown, J., Cherry, F., Most, L., Parker, E., Wallis, W.: Completion of the spectrum of orthogonal diagonal Latin squares. *Lecture notes in pure and applied mathematics* 139, 43–49 (1992)
14. Knuth, D.E.: *The art of computer programming, Volume 4A: combinatorial algorithms*. Addison-Wesley Professional (2013)
15. Vatutin, E., Nikitina, N., Belyshev, A., Manzyuk, M.: On polynomial reduction of problems based on diagonal Latin squares to the exact cover problem. In: Bychkov, I.V., Tchernykh, A., Feoktistov, A.G. (eds.) *Proceedings of the 2nd International Workshop on Information, Computation, and Control Systems for Distributed Environments (ICCS-DE’2020)*. CEUR Workshop Proceedings, vol. 2638, pp. 289–297 (2020)
16. Knuth, D.E.: Dancing links. *Millennial Perspectives in Computer Science* pp. 187–214 (2000)
17. Brayton, R., Coppersmith, D., Hoffman, A.: Self-orthogonal latin squares of all orders $n \neq 2, 3, 6$. *Bulletin of the American Mathematical Society* 80, 116–118 (1974)
18. Vatutin, E., Belyshev, A.: Enumerating the orthogonal diagonal Latin squares of small order for different types of orthogonality. In: Voevodin, V.V., Sobolev, S. (eds.) *Supercomputing - 6th Russian Supercomputing Days, RuSCDays 2020, Moscow, Russia, September 21-22, 2020, Revised Selected Papers. Communications in Computer and Information Science*, vol. 1331, pp. 586–597. Springer (2020)
19. Appa, G., Mourtos, I., Magos, D.: Integrating constraint and integer programming for the orthogonal Latin squares problem. In: Hentenryck, P.V. (ed.) *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings. Lecture Notes in Computer Science*, vol. 2470, pp. 17–32. Springer (2002)
20. Rubin, N., Bright, C., Cheung, K.K.H., Stevens, B.: Integer and constraint programming revisited for mutually orthogonal Latin squares. *CoRR abs/2103.11018* (2021)
21. Bright, C., Gerhard, J., Kotsireas, I.S., Ganesh, V.: Effective problem solving using SAT solvers. In: Gerhard, J., Kotsireas, I.S. (eds.) *Maple in Mathematics Education and Research - Third Maple Conference, MC 2019, Waterloo, Ontario, Canada, October 15-17, 2019, Proceedings. Communications in Computer and Information Science*, vol. 1125, pp. 205–219. Springer (2019)
22. Bogart, K.P.: *Introductory combinatorics*. Harcourt Brace Jovanovich, San Diego, 2nd ed. edn. (1990)
23. Vatutin, E., Belyshev, A., Kochemazov, S., Zaikin, O., Nikitina, N.: Enumeration of isotopy classes of diagonal Latin squares of small order using volunteer computing. In: Voevodin, V.V., Sobolev, S.I. (eds.) *Supercomputing - 4th Russian Supercomputing Days, RuSCDays 2018, Revised Selected Papers. Communications in Computer and Information Science*, vol. 965, pp. 578–586. Springer (2018)

24. Sloane, N.J.A.: The on-line encyclopedia of integer sequences. *Electr. J. Comb.* 1 (1994)
25. Vatutin, E., Titov, V., Zaikin, O., Kochemazov, S., Manzuk, M., Nikitina, N.: Orthogonality-based classification of diagonal Latin squares of order 10. In: *Proceedings of the VIII International Conference on Distributed Computing and Grid-technologies in Science and Education (GRID 2018)*. CEUR Workshop Proceedings, vol. 2267, pp. 282–287 (2018)
26. Vatutin, E., Belyshev, A., Zaikin, O., Nikitina, N., Manzyuk, M.: Investigating properties of generalized symmetries in diagonal Latin squares using volunteer computing (in russian). *High-performance computational systems and technologies* 3(2), 39–51 (2019)
27. Manzyuk, M., Nikitina, N., Vatutin, E.: Start-up and the results of the volunteer computing project RakeSearch. In: *Voevodin, V.V., Sobolev, S. (eds.) Supercomputing - 5th Russian Supercomputing Days, RuSCDays 2019, Moscow, Russia, September 23-24, 2019, Revised Selected Papers*. Communications in Computer and Information Science, vol. 1129, pp. 725–734. Springer (2019)
28. Kochemazov, S., Zaikin, O., Vatutin, E., Belyshev, A.: Enumerating diagonal Latin squares of order up to 9. *J. Integer Seq.* 23(1), 20.1.2 (2020)
29. Vatutin, E., Zaikin, O., Kochemazov, S., Valyaev, S.: Using volunteer computing to study some features of diagonal Latin squares. *Open Engineering* 7, 453–460 (2017)
30. Balyo, T., Sinz, C.: Parallel Satisfiability. In: *Hamadi, Y., Sais, L. (eds.) Handbook of Parallel Constraint Reasoning*, pp. 3–29. Springer (2018)
31. Kochemazov, S., Zaikin, O., Semenov, A.: The comparison of different SAT encodings for the problem of search for systems of orthogonal Latin squares. In: *International Conference Mathematical and Information Technologies - MIT 2016*. CEUR Workshop Proceedings, vol. 1839, pp. 155–165 (2017)
32. Lynce, I., Ouaknine, J.: Sudoku as a SAT problem. In: *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2006, Fort Lauderdale, Florida, USA, January 4-6, 2006* (2006)
33. Biere, A., Fazekas, K., Fleury, M., Heisinger, M.: CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In: *Balyo, T., Froykys, N., Heule, M., Iser, M., Jarvisalo, M., Suda, M. (eds.) Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*. Department of Computer Science Report Series B, vol. B-2020-1, pp. 51–53. University of Helsinki (2020)
34. Heule, M., Dufour, M., van Zwieten, J., van Maaren, H.: March.eq: Implementing additional reasoning into an efficient look-ahead SAT solver. In: *Hoos, H.H., Mitchell, D.G. (eds.) Theory and Applications of Satisfiability Testing, 7th International Conference, SAT 2004, Vancouver, BC, Canada, May 10-13, 2004, Revised Selected Papers*. Lecture Notes in Computer Science, vol. 3542, pp. 345–359. Springer (2004)
35. Kurochkin, I.: The umbrella project of volunteer distributed computing Optima@home. In: *Ivahsko, E., Romyantsev, A. (eds.) Proceedings of the Third International Conference BOINC-based High Performance Computing: Fundamental Research and Development (BOINC:FAST 2017)*. CEUR Workshop Proceedings, vol. 1973, pp. 35–42 (2017)
36. Eén, N., Sörensson, N.: An extensible SAT-solver. In: *Giunchiglia, E., Tacchella, A. (eds.) Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003, Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*. Lecture Notes in Computer Science, vol. 2919, pp. 502–518. Springer (2003)

37. Biró, C., Kovásznai, G., Biere, A., Kuspér, G., Geda, G.: Cube-and-Conquer approach for SAT solving on grids. *Annales Mathematicae et Informaticae* 42, 9–21 (01 2013)
38. Heisinger, M., Fleury, M., Biere, A.: Distributed Cube and Conquer with Paracooba. In: Pulina, L., Seidl, M. (eds.) *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*. Lecture Notes in Computer Science, vol. 12178, pp. 114–122. Springer (2020)
39. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving and verifying the Boolean pythagorean triples problem via Cube-and-Conquer. In: Creignou, N., Berre, D.L. (eds.) *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*. Lecture Notes in Computer Science, vol. 9710, pp. 228–245. Springer (2016)
40. Bright, C., Cheung, K.K.H., Stevens, B., Kotsireas, I.S., Ganesh, V.: A SAT-based resolution of Lam’s problem. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*. pp. 3669–3676. AAAI Press (2021)
41. Fang, W., Beckert, U.: Parallel tree search in volunteer computing: a case study. *J. Grid Comput.* 16(4), 647–662 (2018)
42. Ignatov, A., Posypkin, M.: BOINC-based branch-and-bound. In: Voevodin, V.V., Sobolev, S.I. (eds.) *Supercomputing - 4th Russian Supercomputing Days, RuSC-Days 2018, Moscow, Russia, September 24-25, 2018, Revised Selected Papers*. Communications in Computer and Information Science, vol. 965, pp. 511–522. Springer (2018)
43. Semenov, A., Zaikin, O., Kochemazov, S.: Finding effective SAT partitionings via black-box optimization. In: Pardalos, P.M., Rasskazova, V., Vrahatis, M.N. (eds.) *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, pp. 319–355. Springer International Publishing, Cham (2021)
44. Afanasiev, A.P., Bychkov, I.V., Zaikin, O.S., Manzyuk, M.O., Posypkin, M.A., Semenov, A.A.: Concept of a multitask grid system with a flexible allocation of idle computational resources of supercomputers. *Journal of Computer and Systems Sciences International* 56(4), 701–707 (Jul 2017)
45. Vatutin, E., Kochemazov, S., Zaikin, O.: Applying volunteer and parallel computing for enumerating diagonal Latin squares of order 9. In: *Proc. of The Eleventh International Conference on Parallel Computational Technologies*. Communications in Computer and Information Science, vol. 753, pp. 110–124. Springer (2017)