Investigating the Existence of Orthogonal Golf Designs via Satisfiability Testing

Pei Huang [1,3], Minghao Liu[1,3], Cunjing Ge[1,3], Feifei Ma[1,2,3]*, Jian Zhang[1,3]
[1] State Key Laboratory of Computer Science
[2] Laboratory of Parallel Software and Computational Science
Institute of Software, Chinese Academy of Sciences
[3] University of Chinese Academy of Sciences
Beijing, China
{huangpei,liumh,gecj,maff,zj}@ios.ac.cn

ABSTRACT

A collection of $n-2$ idempotent symmetric quasigroups of order n is called a *golf design* if all the quasigroups in the collection are mutually disjoint. Two golf designs are said to be orthogonal if any idempotent symmetric quasigroup from one golf design has an orthogonal mate in the other golf design, and it is also called an *orthogonal golf design* ($OG(n)$). The existence of orthogonal golf designs is an open problem in combinatorial design theory. In this paper, we describe a method for solving some open cases using automated reasoning tools, employing both symmetry breaking and heuristic decision. The experimental results show that our method is highly efficient and it indeed allowed us to get some positive results in reasonable time. In particular, we apply state-of-the-art SAT solvers and constraint solvers to decide the non-existence of some instances, which can produce a formal proof.

CCS CONCEPTS

• **Mathematics of computing** → **Combinatoric problems**; • **Theory of computation** → *Automated reasoning*.

KEYWORDS

Quasigroup; Combinatorial Design; Orthogonal Golf Designs; Automated Reasoning; Automated Theorem Proving; Logic

ACM Reference Format:
Pei Huang, Minghao Liu, Cunjing Ge, Feifei Ma, Jian Zhang. 2019. Investigating the Existence of Orthogonal Golf Designs via Satisfiability Testing. In *International Symposium on Symbolic and Algebraic Computation (ISSAC '19), July 15–18, 2019, Beijing, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3326229.3326232

1 INTRODUCTION

Combinatorial design [8] has long been the interest of both mathematicians and computer scientists. With a number of new computational approaches appeared, some problems that used to be open have been solved in recent decades. The most notable of these computational approaches is automated reasoning. Euler's conjecture about the existence of orthogonal Latin squares of order 10, which used to be a famous open problem and failed at computer search in 1963 [20], can now be solved by the state-of-the-art automated reasoning tools in tens of seconds. At the end of the last century, many open problems about quasigroups, such as $QG2$ to $QG9$, have been solved by some finite-model generators such as *MGTP, FINDER, SEM, MACE4* and propositional satisfiability provers *SATO, DDPP*, respectively [13, 23, 28–31]. In recent years, Marijn Heule et al. solved the boolean pythagorean triples problem via a parallelized SAT solver with 800 cores in about 2 days [15]. Curtis Bright et al. developed a SAT+CAS paradigm of coupling SAT solvers with computer algebra systems [2, 33], which is capable of enumerating Williamson matrices of even order $n < 65$ [3] and all Golay pairs of length up to 25 [4].

In this paper, we focus on some open problems about the *large set* of quasigroups. The large set problem, which seeks to find a set of combinatorial objects rather than one, is a classic and challenging research topic in combinatorial design theory. The concept of *large set* emerged in the 1850s [6] and the large set of idempotent quasigroups was proposed at the end of 1980s [25]. Due to its difficulty in construction, any progress of the large set is something anticipated [5, 27]. Establishing the existence of some cases of moderate order via computer can provide support for mathematicians to further explore general issues. Sometimes a large object can be produced from smaller ones via mathematical construction. Besides, many hard combinatorial problems related to quasigroups also have potential value in the field of cryptography [18, 22].

A collection of $n-2$ idempotent symmetric quasigroups of order n is called a *large set* if any two of them are disjoint. It is also called a *golf design*. The existence of *golf designs* has been solved by Teirlinck [24], Colbourn and Nonay [9], and Chang [7]. However, the existence of *orthogonal golf designs* (OG) is an open problem. And, it is strongly related to another open problem called *Room square* [21, 26].

We attempt to further study the open cases of OG. These challenging problems are difficult to solve via automated reasoning tools directly, such as SAT solvers, CSP solvers (Constraint Satisfaction Problem solvers) and finite-model generators. Based on our test, no instance can be solved in a week on a personal computer when $n \geq 9$. The tremendous computation burden promotes us to eliminate isomorphic (symmetric) search spaces and seek a more powerful heuristic decision.

ISSAC '19, July 15–18, 2019, Beijing, China

ACM ISBN 978-1-4503-6084-5/19/07…$15.00
https://doi.org/10.1145/3326229.3326232

In order to avoid a lot of symmetric search spaces, we add some symmetry breaking constraints to the model and reformulate the problem. Designing disjoint quasigroups and orthogonal quasigroups are the crux of the problem. The added constraints can reduce the isomorphic situations we identified in disjointness. For orthogonality, we reformulate the orthogonal mate finding problem as the transversal finding problem. Euler originally used this technique to investigate his conjecture in 1779. We modify the technique and make it adapt to the idempotent symmetric quasigroup. We also design a heuristic search procedure which can explore some areas with high priority.

The experimental results show that our method can greatly improve the solving efficiency. Some open cases, which cannot be solved in a week before, can now be solved within a day.

This paper is organized as follows: In Sect. 2, we introduce some preliminaries about the orthogonal golf designs; In Sect. 3 and 4, we describe how to model the problem in logic language and break symmetries; In Sect. 5, we illustrate the heuristic decision and present the search framework; In Sect. 6 and 7, we present the new results we found and the experimental evaluation; In the final section, conclusions are drawn.

2 PRELIMINARIES

2.1 Basic Concepts

A *quasigroup* is denoted as an ordered pair $(Q, \oplus)$, where Q is a set and $\oplus$ is a binary operation on Q. For all constants $a, b \in Q$, equations $a \oplus x = b$ and $y \oplus a = b$ are uniquely solvable. $|Q|$ is said to be the *order* of $(Q, \oplus)$.

For all $x \in Q$, if $x \oplus x = x$ (briefly $x^2 = x$), the quasigroup $(Q, \oplus)$ is *idempotent*.

A *quasigroup* is called *symmetric* if for any $x, y \in Q, x \oplus y = y \oplus x$ (briefly $xy = yx$), We denote an idempotent symmetric quasigroup of order n as $ISQ(n)$.

Two idempotent symmetric quasigroups $(Q, \oplus)$ and $(Q, \odot)$ are called *orthogonal* if for any $u, v \in Q$, $u \neq v$, the equations

$$x \oplus y = u, x \odot y = v$$

either have no solution, or have two solutions.

Two idempotent symmetric quasigroups $(Q, \oplus)$ and $(Q, \odot)$ are said to be *disjoint* if for all $x, y \in Q$, $x \oplus y \neq x \odot y$ whenever $x \neq y$.

A large set of idempotent symmetric quasigroups is a collection of idempotent symmetric quasigroups $\{(Q, \oplus_k) | 1 \leq k \leq n-2)\}$, if any two of them are disjoint. It is also called a *golf design*.

It is known that a pair of two orthogonal $ISQ(n)$ is equivalent to a Room square of side n. And for any odd $n \geq 1$, $n \neq 3, 5$, there exists a Room square of side n.

Definition 2.1. A collection of idempotent symmetric quasigroups $\{(Q, \oplus_k) | 1 \leq k \leq d)\}$ is called Room d-cube of side n if any two idempotent symmetric quasigroups are orthogonal, where $|Q| = n$.

In general, for idempotent quasigroups, orthogonality implies disjointness, but the reverse does not hold.

It is well-known that the multiplication table of a quasigroup is a Latin square. Thus Latin squares and quasigroups are often treated as synonyms. Figure 1 shows the multiplication table of two idempotent symmetric quasigroups where $Q = \{0, 1, 2, 3, 4, 5, 6\}$. They are orthogonal and disjoint.

	0	1	2	3	4	5	6
0	0	2	6	4	5	3	1
1	2	1	4	5	6	0	3
2	6	4	2	0	3	1	5
3	4	5	0	3	1	6	2
4	5	6	3	1	4	2	0
5	3	0	1	6	2	5	4
6	1	3	5	2	0	4	6

	0	1	2	3	4	5	6
0	0	3	4	1	2	6	5
1	3	1	6	0	5	4	2
2	4	6	2	5	0	3	1
3	1	0	5	3	6	2	4
4	2	5	0	6	4	1	3
5	6	4	3	2	1	5	0
6	5	2	1	4	3	0	6

Figure 1: Two orthogonal and disjoint $ISQ(7)$

2.2 The problem

There exists a golf design of order n for any odd $n \geq 3$ with one exception of $n = 5$. However, the existence of orthogonal golf designs is still an open problem.

Definition 2.2. Two golf designs $\{(Q, \oplus_k) | 1 \leq k \leq n-2)\}$ and $\{(Q, \odot_k) | 1 \leq k \leq n-2)\}$ are called orthogonal if for any $1 \leq k \leq n-2$, $(Q, \oplus_k)$ and $(Q, \odot_k)$ are orthogonal.

A pair of orthogonal golf designs of order n is denoted as $OG(n)$, and it can also be viewed as a large set of disjoint Room squares, where the k-th Room square is corresponding to the orthogonal symmetric quasigroups $(Q, \oplus_k)$ and $(Q, \odot_k)$.

We use $OG(n)\_d$ to denote a pair of mutually orthogonal partial golf designs $\{(Q, \oplus_k) | 1 \leq k \leq d)\}$ and $\{(Q, \odot_k) | 1 \leq k \leq d)\}$, where $d \leq n-2$. It is easy to know that Room d-cube implies $OG(n)\_d$. Once the non-existence of some $OG(n)$ is decided, it is still desirable to know the greatest lower bound of d.

The research on Room d-cube has made a lot of progress over the past decades, e.g. Room 4-cube of side 9 was first constructed in 1985 [10] and lower bounds of some cases were summarized in 1992 [11]. However, the least upper bound of d and whether d can reach $n-2$ for many cases are still open problems.

Based on the known lower bounds of some Room cube instances, which can be found in p590 of *Handbook of Combinatorial Designs* [8], the lower bounds for $OG(n)\_d$ can be derived. Table 1 shows the lower bounds for $OG(n)\_d$ of moderate sizes . '=' indicates the least upper bound. '≥' means the best known result.

Table 1: Lower bounds for $OG(n)\_d$ **(**$n \leq 50$**)**

n	d	n	d	n	d	n	d
1	$= 0$	13	≥ 5	25	≥ 7	37	≥ 15
3	$= 0$	15	≥ 4	27	≥ 13	39	≥ 5
5	$= 0$	17	≥ 5	29	≥ 13	41	≥ 9
7	≥ 3	19	≥ 9	31	≥ 15	43	≥ 21
9	≥ 4	21	≥ 5	33	≥ 5	45	≥ 5
11	≥ 5	23	≥ 11	35	≥ 5	47	≥ 23

$n = 7$ can be regarded as the real starting point for $OG(n)$ problems. Our target is to improve the table and investigate whether d can reach $n-2$ for some instances.

3 MODELING

In this section, we will introduce the method to model $OG(n)\_d$ with logic language. Without loss of generality, we assume the domain Q to be the set $\{0, 1, \ldots, n-1\}$. $\oplus_k$ is actually a function $L_k : Q \times Q \mapsto Q$. Similarly, $\odot_k$ is a function $L'_k : Q \times Q \mapsto Q$. $L_k(x, y)$ denotes $x \oplus_k y$ and $L'_k(x, y)$ denotes $x \odot_k y$. We also refer to the cell in position (x, y) of the *Latin square* L_k as $L_k(x, y)$.

Based on the definition of *quasigroup*, it is easy to know that:

$$\begin{aligned} &\forall x \forall y \forall z \ (y = z \vee L_k(x, y) \neq L_k(x, z)) \\ &\forall x \forall y \forall z \ (x = z \vee L_k(x, y) \neq L_k(z, y)) \end{aligned} \tag{1}$$

and

$$\begin{aligned} &\forall x \forall y \forall z \ (y = z \vee L'_k(x, y) \neq L'_k(x, z)) \\ &\forall x \forall y \forall z \ (x = z \vee L'_k(x, y) \neq L'_k(z, y)) \end{aligned}$$

The *symmetric* property ($xy = yx$) can be encoded as:

$$\forall x \forall y \ (L_k(x, y) = L_k(y, x)) \tag{2}$$

and

$$\forall x \forall y \ (L'_k(x, y) = L'_k(y, x))$$

The *idempotent* property ($x^2 = x$) can be encoded as:

$$\forall x \ (L_k(x, x) = x) \tag{3}$$

and

$$\forall x \ (L'_k(x, x) = x)$$

The *disjoint* property depicts that for any two Latin squares L_j and L_k, $L_j(x, y) \neq L_k(x, y)$ except for $x = y$ (L'_j and L'_k as well). So it can be written as:

$$\forall x \forall y \ (x = y \vee L_k(x, y) \neq L_j(x, y)) \tag{4}$$

and

$$\forall x \forall y \ (x = y \vee L'_k(x, y) \neq L'_j(x, y))$$

$(Q, \oplus_k)$ and $(Q, \odot_k)$ are *orthogonal*, based on the definition we know that for all $x1, y1, x2, y2 \in Q$:

$$(x_1 = x_2 \vee L_k(x_1, y_1) \neq L_k(x_2, y_2) \vee L'_k(x_1, y_1) \neq L'_k(x_2, y_2))$$
$$(y_1 = y_2 \vee L_k(x_1, y_1) \neq L_k(x_2, y_2) \vee L'_k(x_1, y_1) \neq L'_k(x_2, y_2))$$

These formulas make up the basic model for the $OG(n)\_d$ problem. However, it is not that efficient for automated reasoning tools and still has room for improvement.

4 IMPROVEMENTS IN MODELING

Arguably, many hard combinatorial problems allow isomorphic solutions, and we say these problems have symmetries. Exploiting symmetry can reduce the search time spent on revisiting equivalent states over and over again when solving the problem. So, it is vital for us to handle the symmetries of the problem at hand. It is common to identify three main approaches to symmetry breaking.

(1) The first method is to add symmetry breaking constraints before search starts, thereby making some symmetric solutions unacceptable while leaving at least one solution in each symmetric equivalence class.
(2) The second is to reformulate the problem so it has a reduced amount of symmetries or make symmetries easy to identify.
(3) The final approach is to break symmetry dynamically during search, adapting the search procedure appropriately.

Although symmetry breaking technique and automatic identification of the symmetry have been concerned by researchers in the past, some latent symmetries still need human intervention.

In this section, we will focus on improving the modeling and reducing symmetries in it.

4.1 Symmetries in Disjointness

First, we examine the structure of the problem and identify symmetries in disjointness.

Lemma 4.1. *If two idempotent symmetric quasigroups $(Q, \oplus_k)$ and $(Q, \odot_k)$ are orthogonal, then they are also orthogonal after any isomorphic permutation σ.*

Lemma 4.2. *If two idempotent symmetric quasigroups $(Q, \oplus_k)$ and $(Q, \oplus_j)$ are disjoint, then they are also disjoint after any isomorphic permutation σ.*

Lemma 4.1 and 4.2 are quite easy to prove, so we will not detail the proofs here.

Proposition 4.3. *If there is an $OG(n)\_d$ consisting of $\{(Q, \oplus_k)|1 \leq k \leq d)\}$, $\{(Q, \odot_k)|1 \leq k \leq d)\}$, then there exists an $OG(n)\_d$ such that $0 \oplus_k 1 = k + 1$ (or $L_k(0, 1) = k + 1$).*

Proof. According to $x^2 = x$, we know that $L_k(0, 0) = 0$ and $L_k(1, 1) = 1$. Since $a \oplus_k x = b$ and $y \oplus_k a = b$ are uniquely solvable, $L_k(0, 1)$ cannot be 0 or 1. All candidates for it include $2, 3, \ldots, n-1$ ($n-2$ elements). Due to the property of *disjoint*, for any $1 \leq k_1, k_2 \leq d$, $L_{k_1}(0, 1) \neq L_{k_2}(0, 1)$. We denote the quasigroup $(Q, \oplus_k)$ as $L^{(j)}$ which $L_k(0, 1) = j + 1$. Without loss of generality we assume $\{(Q, \oplus_k)|1 \leq k \leq d)\}$ is $\{L^{(j_1)}, L^{(j_2)}, \ldots, L^{(j_d)}\}$ where $j_1 < j_2, \ldots, < j_d$. Then, we can construct a permutation σ in Cauchy form:

$$\sigma : \begin{pmatrix} 0 & 1 & 2 & 3 & \ldots & d+1 & d+2 & \ldots & n-1 \\ 0 & 1 & j_1+1 & j_2+1 & \ldots & j_d+1 & * & \ldots & * \end{pmatrix}$$

The '*' can be any legitimate number. Then we can perform σ^{-1} on $\{L^{(j_1)}, L^{(j_2)}, \ldots, L^{(j_d)}\}$ and $\{(Q, \odot_k)|1 \leq k \leq d)\}$. Based on the lemma 4.1 and 4.2, the orthogonality and disjointness still hold. It is easy to know that $\{L^{(j_1)}, L^{(j_2)}, \ldots, L^{(j_d)}\}^{\sigma^{-1}} = \{L^{(1)}, L^{(2)}, \ldots, L^{(d)}\}$. □

Proposition 4.3 reveals that any d disjoint $ISQ(n)$s must be isomorphic to some $\{(Q, \oplus_k)|1 \leq k \leq d)\}$ where $L_k(0, 1) = k + 1$. So we can take advantage of this property and add constraints to fix $L_k(0, 1)$:

$$L_k(0, 1) = k + 1 \tag{5}$$

In this way, $P(n-2, d) - 1$ isomorphic cases can be eliminated, where $P(n-2, d)$ denotes d-permutations of $n-2$.

4.2 Reformulation of Orthogonality

We know that finding a pair of orthogonal Latin squares is equivalent to the transversal-finding phase[17, 19]. If one attempts to use enumeration method searching an orthogonal mate, transversal-finding paradigm will reduce a lot of computation compared with direct finding method. As described by Donald Knuth in Volume 4A

of TAOCP [17] transversal-finding paradigm will reduce a factor of more than 10^{12}(!) when searching Euler's conjecture of order 10. In [19], Feifei Ma et al. show that modeling an orthogonal mate finding problem via transversal-finding paradigm can also improve the efficiency of automated reasoning tools.

However, the original version of transversal-modeling cannot be used directly for this problem, it needs some modification to adapt the idempotent symmetric quasigroups. We will start with the transversal-modeling for ordinary Latin squares and then clarify how to migrate this paradigm to our problems.

Definition 4.4. A transversal in a Latin square is a collection of positions, one from each row and one from each column, so that the elements in these positions are all different.

A transversal in a Latin square L can be written as a vector, where the i-th element records the row index of the cell that appears in the i-th column. For example, in Figure 2, the positions marked by '-' (or '---') make up a transversal of the Latin square L. The transversal marked by '-' is $\{(0,0),(2,1),(4,2),(1,3),(3,4)\}$ and it can be abbreviated as a vector $t =< 0,2,4,1,3 >$. The indexes of t represents the column.

Definition 4.5. Two transversals in a Latin square are said to be disjoint, if their intersection is an empty set $\emptyset$.

In the view of the transversal vector, two transversal vectors t and t' are disjoint means for all i, $t[i] \neq t'[i]$. In Figure 2, two disjoint transversals of L are shown.

Definition 4.6. A matrix T is called a transversal matrix of Latin square L, if T is consisted of n mutually disjoint transversal vectors, where $|Q| = n$.

In Figure 2, T is a transversal matrix of L. It is made up of five disjoint transversal vectors. A transversal matrix is also a Latin square, otherwise it will contradict definition 4.5.

Two ordinary Latin squares L and L' are said to be othogonal, if for all $x_1, x_2, y1, y2 \in Q$, the ordered pair $(L(x_1, y_1), L'(x_2, y_2))$ is unique.

Proposition 4.7. *A Latin square L has an orthogonal mate, iff L has a transversal matrix.*

If L has a transversal matrix, that means it has n disjoint transversals. For each transversal, we can assign a different element. Then, we fill the assigned element in the position where the transversal record and an othogonal mate L' will be constructed. If L has an othogonal mate L', we can extract the positions from L' which have the same element and these will make up a transversal of L. For each element we extract a transversal and these transversals will form a transversal matrix.

Based on Proposition 4.7, finding an orthogonal mate for a Latin square is equivalent to finding a transversal matrix for it.

Actually, transversal matrix can be seen an encoding matrix for the othogonal mate. Since the transversal matrix focus on the positions information rather than the elements themselves, a cluster of isomorphic othogonal mates can be decoded from a transversal matrix. Figure 3 shows that we can generate L' and L'' based on the transversal matrix T. If we assign elements $(0,1,2,3,4)$ to each transversal in order, we can decode L'. When we assign elements $(2,0,4,1,3)$ to each transversal, we can decode L''. They are isomorphic to each other and orthogonal to L in Figure 2.

0	2	4	1	3
1	3	0	2	4
2	4	1	3	0
3	0	2	4	1
4	1	3	0	2

T

0	3	1	4	2
1	4	2	0	3
2	0	3	1	4
3	1	4	2	0
4	2	0	3	1

L'

2	1	0	3	4
0	3	4	2	1
4	2	1	0	3
1	0	3	4	2
3	4	2	1	0

L''

Figure 3: Decode two isomorphic orthogonal mates of L from transversal matrix T

If we interpret the i-th row transversal vector corresponding to the element i, then each transversal matrix can decode only one orthogonal mate. Formally, we describe the relationship between a Latin square L and the transversal matrix T with the logic language as:

$$\forall x \forall y_1 \forall y_2 \ (y_1 = y_2 \vee L(T(x, y_1), y_1) \neq L(T(x, y_2), y_2))$$

Furthermore, we know that after exchanging any two rows of the transversal matrix, it remains to be a transversal matrix. So we can arrange the transversal vectors in transversal matrix to be lexicographically ordered (e.g. fix $T(x,0) = x$) and $(n! - 1)$ isomorphic situations can be avoided. The formula $\forall x T(x,0) = x$ is called a lex-leader constraint.

Finding two orthogonal ISQs is slightly different with ordinary Latin squares. Due to the symmetric property, the ordered pair $(L(x_1, y_1), L'(x_2, y_2))$ is required to be unique in the upper triangular matrix (or the lower). So the concept of the transversal needs to be extended. We call it an ISQ-transversal.

Definition 4.8. An ISQ-transversal in an $ISQ(n)$ is a collection of $(n+1)/2$ positions which are located in upper triangular matrix (or the lower), one from each row and one from each column, so that the elements in these positions are all different.

The ISQ-transversal is a partial transversal of length $(n+1)/2$. We still use an n-dimensional vector to record it and just some positions are empty. These blanks can be filled according to the property of symmetry.

We assume that T_k is an ISQ-transversal matrix of L_k and it can be decoded as L'_k. We interpret the i-th row ISQ-transversal vector corresponding to the element i. Since the L'_k is an idempotent $(x^2 = x)$ quasigroup, we can fix the diagonal of T_k:

$$\forall x \ (T_k(x,x) = x) \tag{6}$$

In this context, the lexicographic order of the ISQ-transversal matrix is fixed.

For $ISQ(n)$, the orthogonality only concerns the upper triangular matrix. We know that the elements in the T_k represent 'row', so the relationship between L_k and T_k is:

$$\begin{aligned} \forall x \forall y_1 \forall y_2 \ (T_k(x, y1) > y1 \vee T_k(x, y2) > y2 \vee \\ y_1 = y_2 \vee L_k(T_k(x, y_1), y_1) \neq L_k(T(x, y_2), y_2)) \end{aligned} \tag{7}$$

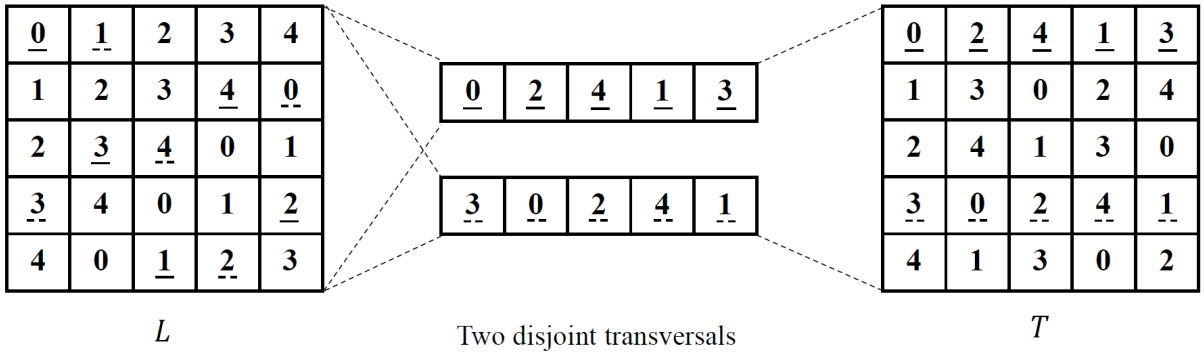

Figure 2: Transversal vectors and transversal matrix of Latin square L

Formula (7) means that if $T_k(x, y1) \leq y1$ and $T_k(x, y2) \leq y2$ then we have $y_1 = y_2 \vee L_k(T_k(x, y_1), y_1) \neq L_k(T_k(x, y_2), y_2))$. $T_k(x, y1) \leq y1$ and $T_k(x, y2) \leq y2$ specify that the elements of interest are taken from the upper triangular matrix.

L'_k is symmetric and this property is embodied in T_k as:

$$\forall x \forall y \; T_k(x, T_k(x, y)) = y \tag{8}$$

Formula (8) help us fill the blanks of each ISQ-transversal vector in the T_k.

It is easy to know that T_k is also a Latin square, otherwise two different elements will occur in the same position of L'_k. So we have:

$$\begin{aligned} &\forall x \forall y \forall z \; (y = z \vee T_k(x, y) \neq T_k(x, z)) \\ &\forall x \forall y \forall z \; (x = z \vee T_k(x, y) \neq T_k(z, y)) \end{aligned} \tag{9}$$

In orthogonal golf designs problem, L'_k and L'_j are disjoint. We can prove it is equivalent to that T_k and T_j are disjoint under the fixed interpretation context. For some x, y and $x \neq y$, if $T_k(x, y) = T_j(x, y) = v$ then we can deduce a conflict that $L_k(v, y) = L_j(v, y) = x$ and vice versa. So we have:

$$\forall x \forall y \; (x = y \vee T_k(x, y) \neq T_j(x, y)) \tag{10}$$

5 THE SEARCH FRAMEWORK

We use formulas (1)~(10) to model the $OG(n)\_d$ problems. The formulas are in logical conjunction. This is the standard input form for mainstream automated reasoning tools and the abbreviation is called *clause set*. A clause set is a set of clauses and represents a conjunction of the clauses in the set. A clause is a set of literals (atoms or their negations) and represents a disjunction of the literals in the set.

Even though a lot of isomorphic situations have been eliminated in this model, few open instances can be solved directly by automated reasoning tools (e.g. SAT solvers, CSP solvers and finite model generators) in a week. Since the search space is exponential, the general search strategies often failed in giving a result within a reasonable time on a personal computer.

In this case, we want to explore some subspaces with high priority. We know that an $OG(n)\_(d+1)$ should be extended from a known $OG(n)\_d$. So, once we find an $OG(n)\_d$, we try to improve it to $OG(n)\_(d+1)$ at first and exhaust the subspaces expanded from $OG(n)\_d$.

Algorithm 1: The search framework for $OG(n)$

```
Input: n, lb
Output: Maxd, MaxModel
for d ← lb to n − 2 do                          /* Initialization */
    F_d ← Γ_d;
    Model(n)_d ← UNSAT ;
end
Maxd ← 0;
MaxModel ← UNSAT;
while d ≥ lb do
    F_d ← F_d ∪ ¬Model(n)_d ;
    Model(n)_d ← ARsolver(F_d) ;
    if Model(n)_d ≠ UNSAT then
        if d > Maxd then
            Maxd ← d ;
            MaxModel ← Model(n)_d ;
        end
        F_{d+1} ← Γ_{d+1} ∪ Model(n)_d ;        /* Fix OG(n)_d */
         d ← d + 1 ;
    else
        F_d ← Γ_d ;
        d ← d − 1;                              /* Backtracking */
    end
end
return Maxd and MaxModel
```

Algorithm 1 is the search framework. The variable lb denotes the greatest lower bound we know. The variable $Maxd$ is the largest d the process has found and $MaxModel$ stores the corresponding model. $ARsolver()$ denotes an automated reasoning solver. If $Maxd$ becomes greater than lb for some n in search process, it means that we find a better lower bound for this instance. Γ_d denotes the initial clause-set consisting of (1)~(10). In logic, the model can also be denoted as a clause-set, so we use $Model(n)\_d$ to denote the model clause-set and $\overline{Model(n)\_d}$ to represent its negation. $UNSAT$ means empty set or $\{False\}$ and $\overline{UNSAT}$ is $\{True\}$. $F_d \leftarrow F_d \cup \overline{Model(n)\_d}$ means we add the negation of model clause-set to F_d. It can remove the model found by the solver in the previous iteration from the

solution space of F_d. The operation $F_{d+1} \leftarrow \Gamma_{d+1} \cup Model(n)\_d$ generates the clause-set of $OG(n)\_d + 1$ which is trying to expand from the $OG(n)\_d$ found just now.

We choose the CSP solver as the core automated reasoning engine, because formula (1) and (9) can be optimized as the global constraint 'Alldifferent', and a lot of works are devoted to improving the reasoning efficiency of 'Alldifferent' constraints for CSP solvers [1, 12, 32].

In this framework, one can choose a state-of-the-art SAT solver as a substitute for the CSP solver. Due to the finite domain Q, these formulas can also be translated to propositional logic (SAT) formulas. The translation method can be found in [16] and it will be omitted in this paper. Based on our experiments, there is no essential difference between them for this problem. Nevertheless, the SAT solver cannot help us get more results, and it is slightly slower than the CSP solver in the solving process.

Actually, the search process hardly ever terminates in a reasonable time on a personal computer. However, an improved *Maxd* can be found in a reasonable time.

6 NEW RESULTS

With the help of symmetry breaking and search framework, the efficiency for finding $OG(n)\_d$ is improved. We found some new results and list them in Table 2. The details about $OG(13)\_6$, $OG(15)\_7$ and $OG(17)\_7$ are listed in (http://www.square16.org/automatedreasoning/op/).

For $n = 7$, we prove that $OG(7)\_4$ does not exist via SAT solver and the greatest d is equal to 3. The state-of-the-art SAT solver supports emission of a standard unsatisfiability proof which can be verified by a checker [14, 15].

Table 2: New results

Order n	Original results	New results
7	$d \geq 3$	$d = 3$
13	$d \geq 5$	$d \geq 6$
15	$d \geq 4$	$d \geq 7$
17	$d \geq 5$	$d \geq 7$

Some mathematicians conjecture that for every odd $n \geq 7$, the least upper bound of d is $(n-1)/2$ for the Room d-cube problem. One of the fact is Room d-cube of side n implies $OG(n)\_d$. So we also conjecture that $d = (n-1)/2$ for $OG(n)$ problems. The existence of some $OG(n)\_(n-1)/2$ of small order such as $OG(7)\_3$, $OG(9)\_4$, $OG(11)\_5$, $OG(13)\_6$, $OG(15)\_7$ are consistent with the conjecture.

It is noteworthy that if one can prove the least upper bound of d is $(n-1)/2$ for OG problem, then one can conclude that it is also the least upper bound for Room d-cube problem.

7 EXPERIMENTAL EVALUATION

In this section, we evaluate the efficiency of our method on a range of OG problems. The experiments are performed on a Dell laptop with Intel(R) Core (TM) i7-6700 CPU (3.40GHz), operating system Ubuntu 16.04 and 16G memory.

The acceleration effect of adding symmetry breaking constraints for disjointness is conclusive. So the experimental evaluation for it will be omitted. Our experiments evaluate mainly the effectiveness of the revised transversal-finding technique and heuristic decision.

7.1 The Benefit of Transversal Modeling

The transversal-finding paradigm is used to accelerate the orthogonal mate finding. We compare it with the straightforward method. Both methods add symmetry breaking constraints for disjointness, and the only difference is that one uses the transversal modeling while the other does not.

In the experiments, we try to find some $OG(n)\_d$ where d is not large for the straightforward method. When $d = 1$, it is equivalent to finding two mutually orthogonal $ISQ(n)$s. In order to eliminate the effect of solvers, we conduct experiments on different kinds of automated reasoning tools representing the state-of-the-art in their own respective categories. *Minizinc* is a CSP solver, and *Glucose* is a SAT solver which is based on *Minisat* and has won many awards in recent SAT competitions. The results are shown in Table 3.

We can see that the transversal-finding paradigm significantly improves the solving efficiency. It has obvious acceleration effect for different kinds of solvers. With the increase of n and d, the advantage of this method becomes more and more remarkable. Based on the experiments, we found that the CSP solver is more suitable for solving this problem. So we choose the CSP solver as the core engine to tackle the open cases.

7.2 The Benefit of Search Framework

Without the help of the search framework, only $OG(15)\_6$ and $OG(7)\_4$ can be solved in a day with the symmetry breaking technique. We failed in searching the other open instances in a week.

Algorithm 1 searches some subspaces with high priority and the results show that it is really effective for these problems. Algorithm 1 never terminates within a week in our experiments except for $n = 7$. But, once the variable *Maxd* is updated, it means that we find a new result. We recorded the time interval between two successive *Maxd* updates. The experimental data are listed in Table 4. The total time to find $OG(13)\_6$, $OG(15)\_7$ and $OG(17)\_7$ are shown in the last column.

Table 4: The run time for updating *Maxd*

n	$d = 3$	$d = 4$	$d = 5$	$d = 6$	$d = 7$	$d = 8$	Total
7	0.8s	-	-	-	-	-	-
9	-	8.7h	>1w	-	-	-	-
11	-	-	8.6h	>1w	-	-	-
13	-	-	2.6m	1.6h	>1w	-	1.64h
15	-	33.2s	34.7m	1.6h	19.5h	>1w	21.67h
17	-	-	2.5h	10.2h	44.8m	>1w	13.44h

The instances $OG(13)\_6$, $OG(15)\_7$ and $OG(17)\_7$ which cannot be solved in a week can now be tackled in a day, demonstrating the effectiveness of the heuristic decision. Besides, the general trend is that the closer the d is to $(n-1)/2$, the harder it is to solve.

Table 3: The run times of different methods in solving $OG(n)\_d$

Instance	Minizinc		Glucose 4.1		Glucose (Parallel)	
	Transversal	Straightforward	Transversal	Straightforward	Transversal	Straightforward
$OG(7)\_1$	0.23s	0.38s	0.02s	0.02s	0.01s	0.04s
$OG(7)\_2$	0.34s	0.45s	0.03s	0.03s	0.03s	0.08s
$OG(7)\_3$	0.35s	2.32s	0.19s	0.14s	0.35s	0.07s
$OG(7)\_4$	114s	143.54s	35.98s	104.52s	10.97s	37.65s
$OG(9)\_1$	0.26s	0.51s	0.12s	0.12s	0.08s	0.16s
$OG(9)\_2$	0.36s	0.81s	0.25s	1.07s	0.19s	0.85s
$OG(9)\_3$	12.20s	86.24s	6.48s	565.69s	1.62s	440.05s
$OG(11)\_1$	0.44s	0.73s	0.49s	2.49s	0.40s	0.54s
$OG(11)\_2$	0.60s	2.50s	1.33s	11683.45s	0.80s	6882.6
$OG(11)\_3$	2.74s	10.37s	3384.58s	>1 day	182.84s	>1 day
$OG(11)\_4$	2754.34s	10279.00s	>1 day	>1 day	>1 day	>1 day
$OG(13)\_1$	1.91s	4.74s	1.90s	11.44s	2.13s	128.66s
$OG(13)\_2$	2.31s	25.13s	209.84s	>1 day	57.46s	>1 day
$OG(13)\_3$	4.72s	321.45s	>1 day	>1 day	>1 day	>1 day

8 CONCLUSIONS

This paper describes an application of automated reasoning techniques and tools to an interesting problem in combinatorics: the orthogonal golf designs (OG). The OG of moderate orders which are difficult for mathematical methods can also be quite challenging for computers. We present some effective solving strategies for this problem: symmetry breaking and heuristic decision. As a result, we find a number of new instances and prove the least upper bound of $OG(7)$. Beside, we conjecture that the least upper bound of $OG(n)$ is $(n-1)/2$, which is consistent with the conjecture of another open problem with strong relevance called Room d-cube.

ACKNOWLEDGMENTS

This work has been supported in part by the Key Research Program of Frontier Sciences, Chinese Academy of Sciences (CAS), Grant No. QYZDJ-SSW-JSC036. Feifei Ma is also supported by the Youth Innovation Promotion Association, CAS. We thank professor Lie Zhu at SooChow university for suggesting these open problems and his valuable advice.

REFERENCES

[1] Christian Bessiere, George Katsirelos, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. 2010. Propagating Conjunctions of AllDifferent Constraints. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010.*

[2] Curtis Bright, Vijay Ganesh, Albert Heinle, Ilias S. Kotsireas, Saeed Nejati, and Krzysztof Czarnecki. 2016. MathCheck2: A SAT+CAS Verifier for Combinatorial Conjectures. In *Computer Algebra in Scientific Computing - 18th International Workshop, CASC 2016, Bucharest, Romania, September 19-23, 2016, Proceedings.* 117–133.

[3] Curtis Bright, Ilias S. Kotsireas, and Vijay Ganesh. 2018. A SAT+CAS Method for Enumerating Williamson Matrices of Even Order. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018.* 6573–6580.

[4] Curtis Bright, Ilias S. Kotsireas, Albert Heinle, and Vijay Ganesh. 2018. Enumeration of Complex Golay Pairs via Programmatic SAT. In *Proceedings of the 2018 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2018, New York, NY, USA, July 16-19, 2018.* 111–118.

[5] H Cao, L Ji, and L Zhu. 2004. Large sets of disjoint packings on 6k+ 5 points. *Journal of Combinatorial Theory, Series A* 108, 2 (2004), 169–183.

[6] Arthur Cayley. 1850. IV. On the triadic arrangements of seven and fifteen things. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 37, 247 (1850), 50–53.

[7] Yanxun Chang. 2007. The existence spectrum of golf designs. *Journal of Combinatorial Designs* 15, 1 (2007), 84–89.

[8] Charles J Colbourn and Jeffrey H Dinitz. 2006. *Handbook of combinatorial designs.* CRC press.

[9] Charles J Colbourn and Gillian Nonay. 1997. A golf design of order 11. *Journal of statistical planning and inference* 58, 1 (1997), 29–31.

[10] JH Dinitz and WD Wallia. 1985. Four orthogonal one-factorizations on ten points. In *North-Holland mathematics studies.* Vol. 114. Elsevier, 143–149.

[11] Jeffrey H Dinitz and Douglas R Stinson. 1992. Room squares and related designs. *Contemporary Design Theory: A collection of surveys* (1992), 137–204.

[12] Michael R. Fellows, Tobias Friedrich, Danny Hermelin, Nina Narodytska, and Frances A. Rosamond. 2011. Constraint Satisfaction Problems: Convexity Makes AllDifferent Constraints Tractable. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011.* 522–527.

[13] Masayuki Fujita, John K. Slaney, and Frank Bennett. 1993. Automatic Generation of Some Results in Finite Algebra. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993.* 52–59.

[14] Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. 2013. Verifying Refutations with Extended Resolution. In *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings.* 345–359.

[15] Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek. 2016. Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer. In *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings.* 228–245.

[16] Pei Huang, Feifei Ma, Cunjing Ge, Jian Zhang, and Hantao Zhang. 2018. Investigating the Existence of Large Sets of Idempotent Quasigroups via Satisfiability Testing. In *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings.* 354–369.

[17] Donald E Knuth. 2011. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 2.* Pearson Education India.

[18] Aleksandar Krapez. 2010. An application of quasigroups in cryptology. *Math. Maced* 8 (2010), 47–52.

[19] FeiFei Ma and Jian Zhang. 2013. Finding orthogonal latin squares using finite model searching tools. *Science China Information Sciences* 56, 3 (2013), 1–9.

[20] E.T. Parker. 1963. Computer investigation of orthogonal Latin squares of order ten. In *Proc. Sympos. Appl. Math*, Vol. 15. 73–81.

[21] KR Shah. 1970. Analysis of Room's square design. *The Annals of Mathematical Statistics* (1970), 743–745.

[22] Victor A Shcherbacov. 2010. Quasigroups in cryptology. *arXiv preprint arXiv:1007.3572* (2010).

[23] John Slaney, Masayuki Fujita, Mark Stickel, et al. 1995. Automated reasoning and exhaustive search: Quasigroup existence problems. *Computers and Mathematics with Applications* 29, 2 (1995), 115–132.
[24] Luc Teirlinck. 1990. On the use of pairwise balanced designs and closure spaces in the construction of structures of degree at least 3. *Le Matematiche* 45, 1 (1990), 197–218.
[25] Luc Teirlinck and CC Lindner. 1988. The construction of large sets of idempotent quasigroups. *European Journal of Combinatorics* 9, 1 (1988), 83–89.
[26] WD Wallis. 1974. Solution of the Room square existence problem. *Journal of Combinatorial Theory, Series A* 17, 3 (1974), 379–383.
[27] Landang Yuan and Qingde Kang. 2008. Some infinite families of large sets of Kirkman triple systems. *Journal of Combinatorial Designs* 16, 3 (2008), 202–212.
[28] Hantao Zhang. 1997. SATO: An Efficient Propositional Prover. In *Automated Deduction - CADE-14, 14th International Conference on Automated Deduction, Townsville, North Queensland, Australia, July 13-17, 1997, Proceedings*. 272–275.
[29] Hantao Zhang. 2009. Combinatorial Designs by SAT Solvers. In *Handbook of Satisfiability*. 533–568.
[30] Hantao Zhang and Mark E. Stickel. 2000. Implementing the Davis-Putnam Method. *J. Autom. Reasoning* 24, 1/2 (2000), 277–296.
[31] Jian Zhang and Hantao Zhang. 1995. SEM: a System for Enumerating Models. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*. 298–303.
[32] Xizhe Zhang, Qian Li, and Weixiong Zhang. 2018. A Fast Algorithm for Generalized Arc Consistency of the Alldifferent Constraint. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. 1398–1403.
[33] Edward Zulkoski, Curtis Bright, Albert Heinle, Ilias S. Kotsireas, Krzysztof Czarnecki, and Vijay Ganesh. 2017. Combining SAT Solvers with Computer Algebra Systems to Verify Combinatorial Conjectures. *J. Autom. Reasoning* 58, 3 (2017), 313–339.