Investigating the Existence of Costas Latin Squares via Satisfiability Testing

Jiwei Jin[1,4], Yiqi Lv[1,3], Cunjing Ge[5], Feifei Ma[1,2,3(✉)], and Jian Zhang[1,3]

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
maff@ios.ac.cn
[2] Laboratory of Parallel Software and Computational Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
[4] Shandong Jiaotong University, Jinan, China
[5] Johannes Kepler University Linz, Linz, Austria

Abstract. Costas Latin squares are important combinatorial structures in combinatorial design theory. Some Costas Latin squares are found in recent years, but there are still some open problems about the existence of Costas Latin squares with specified properties including idempotency, orthogonality, and certain quasigroup properties. In this paper, we describe an efficient method for solving these problems using state-of-the-art SAT solvers. We present new results of Costas Latin squares with specified properties of even order $n \leq 10$. It is found that within this order range, most Costas Latin squares with such properties don't exist except for a few cases. The non-existence can be certified since SAT solvers can produce a formal proof. Experimental results demonstrate the effectiveness of our method.

1 Introduction

Costas Latin squares (introduced in [4]) are important combinatorial structures which have potential applications in industries [2]. The existence of Costas Latin squares were studied in recent years. J. Dinitz et al. [3] studied Costas Latin squares from a construction as well as a classification point of view, and verified the conjecture that there is no Costas Latin square for any odd order $n>3$. Costas Latin squares which have specified properties are also the interest of mathematicians. These properties include idempotency, orthogonality, and certain quasigroup properties, which are often used as the basis of recursive construction. The existence problems of these Costas Latin squares are still open problems, and are difficult for conventional mathematical methods.

With the rapid advance in SAT solving techniques, some problems, which used to be very difficult for traditional mathematic methods, have been resolved recently by SAT solvers. Marijn Heule et al. solved some long-standing open problem, such as the boolean pythagorean triples problem and Schur Number

C.-M. Li and F. Manyà (Eds.): SAT 2021, LNCS 12831, pp. 270–279, 2021.
https://doi.org/10.1007/978-3-030-80223-3_19

Five problem via parallel SAT solving techniques [5,6]. Curtis Bright et al. developed a SAT+CAS paradigm of coupling SAT solvers with computer algebra systems [1,12], which has tackled various combinatorial problems. SAT solving techniques also play an important role in the study of quasigroups [11]. Pei Huang et al. described a method for solving the large set problem of idempotent quasigroups [8].

In this paper, we focus on open problems about Costas Latin squares with specified properties. We attempt to find instances of Costas Latin squares with those properties, or to decide the non-existence of them if they don't have the specified properties via state-of-the-art SAT solvers.

Searching for Costas Latin squares with specified properties are quite challenging for computers. In this paper we present two effective solving strategies: the transversal matrix and symmetry breaking. The transversal matrix is used to reduce the complexity in modeling, while symmetry breaking is used to prune isomorphic search spaces. Experiments are conducted to test the effectiveness of our strategies. The results show that both strategies are highly efficient.

Since Costas Latin squares of order $n \leq 3$ are trivial or simple, and it is conjectured that there is no Costas Latin square for any odd order $n > 3$ [3], we only focus on Costas Latin squares of order $4, 6, 8, 10$. We derive new results for Costas Latin squares of this order range with aforementioned properties, including some instances of Costas Latin squares with certain properties and the non-existence of most cases. The newly discovered Costas Latin squares have been double checked with another program we developed, and the non-existence results can be validated thanks to the capability of modern SAT solvers.

This paper is organized as follows: In Sect. 2, we introduce some preliminaries about Costas Latin squares; In Sect. 3, 4 we describe how to model these problems in logic language and techniques used for improving the searching efficiency; In Sect. 5 we present the new mathematical results and experimental results with analysis; In the final section, we give conclusions.

2 Preliminaries

A Latin square is a $n \times n$ array filled with n different symbols, each occurring exactly once in each row and exactly once in each column. In this paper we used the integer sequence $1, 2, 3, \cdots, n$ as symbols.

A Costas array of order n is a $n \times n$ array of dots and empty cells such that: (a). There are n dots and $n \times (n-1)$ empty cells, with exactly one dot in each row and column. (b). All the segments between pairs of dots differ in length or in slope.

For notational convenience, Costas arrays are often presented by a certain one-line notation. For a Costas array of order n, we use $\pi(i) = j$ whenever a dot is in cell(i,j). By this notation, a Costas array of order n can be presented as the permutation $(\pi(1), \pi(2), \cdots, \pi(n))$.

A Costas Latin square of order n is a Latin square of order n such that for each symbol $i \in \{1, 2, \cdots, n\}$, a Costas array results if a dot is placed in the cells containing symbol i. We use *CLS(n)* to denote Costas Latin square of order n. The follows is a *CLS(4)*:

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

The one-line representation of the symbol 1 is $\{1, 3, 4, 2\}$, 2 is $\{2, 1, 3, 4\}$, 3 is $\{4, 2, 1, 3\}$, 4 is $\{3, 4, 2, 1\}$. They are all Costas array, so the Latin square is a Costas Latin square.

For a *CLS(n)* A, we use $A(i, j)$ to denote the symbol in the i-th row and the j-th column. If A has the property that $A(i, i) = i$ for all $i \in \{1, 2, \cdots, n\}$, then it is called an idempotent Costas Latin square.

The orthogonality is an interesting property of Latin squares. For two *CLS(n)* A and B, if for all $n \times n$ positions, the pair $(A(i, j), B(i, j)), i, j \in \{1, 2, \cdots, n\}$ are different, then A and B are called orthogonal. The follows are two orthogonal *CLS(4)* and the result pairs:

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

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

24	33	42	11
43	14	21	32
31	22	13	44
12	41	34	23

A quasigroup is an algebraic structure such that the multiplication table of a finite quasigroup is a Latin square. Conversely, every Latin square can be taken as the multiplication table of a quasigroup. The existence of quasigroups satisfying the seven short identities has been studied systematically. These identities are:

- 1. $xy \otimes yx = x$: Schrӧder quasigroup
- 2. $yx \otimes xy = x$: Steins third law
- 3. $(xy \otimes y)y = x$: C_3-quasigroup
- 4. $x \otimes xy = yx$: Steins first law; Stein quasigroup
- 5. $(yx \otimes y)y = x$
- 6. $yx \otimes y = x \otimes yx$: Steins second law
- 7. $xy \otimes y = x \otimes xy$: Schrӧders first law

If we take a Costas Latin square as a multiplication table of a quasigroup, and it has one of the quasigroup properties mentioned above, then it is called the Costas Latin square with the specified quasigroup property. One task of this paper is to search for Costas Latin square with certain quasigroup property.

3 Modeling

In this section, we will introduce the method to model Costas Latin squares with logic language. We assume that the symbols (numbers) of *CLS* is an integer sequence $1, 2, \cdots, n$, and row index and column index begin with 1. For convenient we use N to denote the set $\{1, 2, \cdots, n\}$.

Since in a Latin square A, each number occurs exactly once in each row and exactly once in each column, it is easy to know that:

$$\begin{aligned} &\forall x, y, x_1, x_2, y_1, y_2 \in N : \\ &x_1 \neq x_2 \mapsto A(x_1, y) \neq A(x_2, y) \\ &y_1 \neq y_2 \mapsto A(x, y_1) \neq A(x, y_2) \end{aligned} \tag{1}$$

For a *CLS(n) A*, the Costas property requires that for each $i \in N$, all the segments between pairs of i differ in length or in slope. That is for all positions with the same number: a) Each four positions don't form a parallelogram. B) If three or four positions are in a line, the distances between them are different. This can be encoded as:

$$\begin{aligned} &\forall x, y, x', y', u, v, u', v' \in N : \\ &(A(x, y) = A(x', y') = A(u, v) = A(u', v') \wedge (x - x' = u - u') \wedge (y - y' = v - v')) \\ &\mapsto x = u \vee x = x' \end{aligned} \tag{2}$$

The orthogonality property involves two *CLS(n) A, B*. This property requires that in all $n \times n$ positions, the pair $(A(i, j), B(i, j)), i, j \in N$ are different. It can be encoded as:

$$\begin{aligned} &\forall x_1, x_2, y_1, y_2 \in N : \\ &x_1 \neq x_2 \mapsto A(x_1, y_1) \neq A(x_2, y_2) \vee B(x_1, y_1) \neq B(x_2, y_2) \\ &y_1 \neq y_2 \mapsto A(x_1, y_1) \neq A(x_2, y_2) \vee B(x_1, y_1) \neq B(x_2, y_2) \end{aligned} \tag{3}$$

The idempotency property of *a CLS(n) A* can be encoded simply as:

$$\forall x \in N : A(x, x) = x \tag{4}$$

The quasigroup properties are easy to be encoded, for example, the formula for the first one is: $\forall x, y \in N : A\big(A(x, y), A(y, x)\big) = x$. Due to length limitation we omit the logic formulas for them.

4 Improvements in Modeling

Since the basic models in Sect. 3 are hard for SAT solvers, we introduce some search strategies to improve them. The most important strategies are symmetry breaking and the transversal matrix. We say that combinatorial problems have symmetries if they allow isomorphic solutions. Symmetry breaking can reduce the search time spending on revisiting equivalent states of these problems, and is used widely in search algorithms. In this paper we propose a simple but effective symmetry breaking method. For a *CLS(n) A*, all numbers in it are just symbols, after replacing $1, 2, \cdots, n$ by any its permutation, it is still a Costas Latin square. So the method to break symmetries for Costas Latin squares is just to fix its first column:

$$\forall x \in N : A(x, 1) = x \tag{5}$$

It is easy to see that for *CLS(n)*, the simple symmetry breaking method can reduce the search space by *n!*.

The formula for Costas property and the formula for orthogonality property are difficult to handle. We use a method called transversal finding paradigm to improve the search efficiency. As described by Donald Knuth in [9], transversal-finding paradigm will reduce a factor of more than 1012(!) when searching Eulers conjecture of order 10. In [7,10], the authors show that modeling an orthogonal mate finding problem via transversal-finding paradigm can improve the efficiency of automated reasoning tools. In this paper we will show that the transversal-finding paradigm can also be used for modeling Costas property and improve the solving efficiency.

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. It 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. A matrix is called a transversal matrix of Latin square, if it consists of n mutually disjoint transversal vectors. In this paper, we use a variation of transversal matrix. For a Latin square A of order n, we construct a matrix *TA* for it by this way:

If *A(i,j)=k*, then *TA(k,j)=i*, where $i, j, k \in N$.

We can see that each Latin square has a unique transversal matrix, and each transversal matrix belongs to only one Latin square. Here is an example of a Latin square of order 4 (left) and its transversal matrix (right):

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

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

In transversal matrix, the i-th row represents the row index of i in the original Latin square. Let's see the number 2 of the above example. It is in the 2,1,3,4 row (from left to right) in the original Latin square. In the transversal matrix, this is recorded as the vector of the 2-th row $(2, 1, 3, 4)$.

If a Latin square A doesn't have Costas property, then there is a number i in it and at least two segments between i are same in length and in slope. That is the four points decide a parallelogram. Suppose that the four points are $A(x_1, y_1), A(x_2, y_2), A(x_3, y_3), A(x_4, y_4)$. In order to model Costas property we should consider all the four points in a $n \times n$ matrix. Since in *TA*, the row information of A are collected in the i-th row, $TA(i, y_1) = x_1, TA(i, y_2) = x_2, TA(i, y_3) = x_3, TA(i, y_4) = x_4$, the formula for Costas property involves only the i-th row of TA, a vector of n.

If a Latin square A doesn't have Costas property, then in some i-th row of its transversal matrix *TA*, there must be four column x, y, u, v, which make the following hold:

$TA(i, x) - TA(i, y) = TA(i, u) - TA(i, v) \Leftrightarrow x - y = u - v$

We use transversal matrix to simplify the formula for Costas property by replacing Formula 2 to the following formula for transversal matrix TA:

$$\forall x, y, z, u, v \in N: \\ TA(x, u) - TA(x, y) = TA(x, v) - TA(x, z) \vee u - y = v - z \mapsto y = z \vee u = y \tag{6}$$

Next we will use transversal matrix to reformulate Formula 3 for orthogonality property. From [10], we know that finding a pair of orthogonal Latin squares is equivalent to the transversal-finding phase. The transversal matrix focus on the positions information rather than the elements themselves. For two *CLS(n)* *A, B*, their transversal matrix are *TA, TB* respectively. If we formulate orthogonality property using A, B directly, then we should consider all of the $2 \times n \times n$ positions of A, B. By using transversal matrix TA, TB, we can reduce the formula to involving only $2 \times n$ vectors. For some u and v in *TA, TB*, obviously for a column x, $A(TA(u, x), x) = u$ and $B(TB(v, x), x) = v$, since $TA(u, x)$ and $TB(v, x)$ are row index of u, v in A, B. If $TA(u, x) = TB(v, x)$, then the positions of u, v in A, B are the same, denoted as p_1, and in this cell, the pair $(A(TA(u, x), x), B(TB(v, x), x))$ is (u, v). Suppose that A and B are orthogonal. If there is another position p_2 for A, B in y-th column, and the pair in it is also (u, v), then p_2 must be same as p_1, and $x = y$ holds. Otherwise A and B can't be orthogonal. So we use the following formula for transversal matrix:

$$\forall x, y, u, v \in N : x \neq y \mapsto TA(u, x) \neq TB(v, x) \vee TA(u, y) \neq TB(v, y) \tag{7}$$

Due to length limitation we don't give more formal definitions of transversal matrix. More formal definitions and details are in [10].

5 New Results and Experimental Evaluation

In this section, we derive new results of our methods on Costas Latin squares of order 4,6,8,10 with aforementioned properties. We find some instances of Costas Latin squares with certain properties and decide the non-existence of most cases. The newly discovered Costas Latin squares have been double checked with another program we developed, and the non-existence results can be validated thanks to the capability of modern SAT solvers. Also we evaluate the efficiency of symmetry breaking and transversal matrix strategies. The experiments are performed on a PC with Intel CPU (1.60 GHz), 4G memory, Ubuntu 18.04. We encode Costas Latin square problems as CNF formulas, and solve them by a SAT solver: Glucose with default setting.

5.1 New Results

Table 1 indicates whether CLS(n) with specified properties exist or not. Here *Ide* means idempotency, *Ort* means orthogonality. s(sat) means that the CLS(n)

Table 1. The overall results of searching for $CLS(n)$

Order n	Ide	Quasigroup							Ort
		.1	.2	.3	.4	.5	.6	.7	
$CLS(4)$	s	s	s	s	s	u	u	s	s
$CLS(6)$	u	u	u	u	u	u	u	u	u
$CLS(8)$	s	u	u	u	u	u	u	u	u
$CLS(10)$	u	u	u	u	u	u	u	u	*

with the certain property exists and u (unsat) means they don't exist. $CLS(10)-Ort$ is very hard and isn't solved within 12 h.

For *CLS(4)*, there is no instance with quasigroup identities 5 and 6, and all other properties hold. For CLS of order 6,8,10, only the instances with idempotency exist for CLS(8). Here are some instances:

CLS(4)-Ide

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

CLS(4)-Q1

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

CLS(4)-Q2

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

CLS(4)-Q3

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

CLS(4)-Q4

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

CLS(4)-Q7

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

CLS(4)-Ort-A

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

CLS(4)-Ort-B

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

CLS(8)-Ide

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

5.2 Experimental Evaluation

In order to evaluate the effectiveness of symmetry breaking and transversal matrix strategies, we compared the running times of algorithms using these strategies against those lack one or two strategies. Since the symmetry breaking strategy is not fit for quasigroup identities and idempotent because it may conflict with these properties, we conducted two groups of experiments. One is for the problems to which this strategy is applicable, as illustrated in Table 2. The other is for problems to which it is not applicable, as shown in Table 3.

Table 2 and Table 3 show the running times (in seconds) of different methods in solving CLS problems. We set the timeout to 3600 s. Each column is the running times of various methods for certain problems. $SB + Tr$ means

Table 2. The run times of different methods in solving CLS-Ord and CLS-Ort

	SB+Tr	SB	Tr	non		SB+Tr	SB	Tr	non
CLS(6)-Ord	**0.07**	0.08	**0.07**	0.10	CLS(8)-Ord	**1.39**	100.04	26.46	2207.91
CLS(6)-Ort	**0.28**	2.23	TO	TO	CLS(8)-Ort	**67.04**	1230.96	TO	TO

Table 3. The run times of different methods in solving CLS-Ide and CLS-Qi

	Tr	non		Tr	non		Tr	non
CLS(6)-Ide	**0.07**	0.10	CLS(8)-Ide	**0.97**	17.79	CLS(10)-Ide	**406.59**	TO
CLS(6)-Q1	**0.07**	0.13	CLS(8)-Q1	**1.84**	36.58	CLS(10)-Q1	**351.70**	TO
CLS(6)-Q2	**0.08**	0.19	CLS(8)-Q2	**2.88**	97.45	CLS(10)-Q2	**889.16**	TO
CLS(6)-Q3	**0.07**	0.10	CLS(8)-Q3	**1.00**	2.16	CLS(10)-Q3	**12.05**	38.61
CLS(6)-Q4	**0.07**	0.09	CLS(8)-Q4	**0.98**	1.86	CLS(10)-Q4	**10.93**	36.37
CLS(6)-Q5	**0.09**	0.12	CLS(8)-Q5	**3.73**	5.83	CLS(10)-Q5	880.58	**84.13**
CLS(6)-Q6	**0.07**	0.10	CLS(8)-Q6	**0.94**	6.68	CLS(10)-Q6	**11.06**	TO
CLS(6)-Q7	**0.07**	0.10	CLS(8)-Q7	**1.01**	2.21	CLS(10)-Q7	**12.09**	TO

that the method employs both symmetry breaking and transversal matrix; *SB* and *Tr* mean that methods using only symmetry breaking and only transversal matrix respectively. *non* means that the method uses neither of these two strategies. *CLS(i)-Ord* represents ordinary $CLS(i)$, i.e., $CLS(i)$ without any property. Although the existential problems of *CLS(i)-Ord* have been determined by other work, we still use them to evaluate our strategies. $Q1, \cdots, Q7$ represent the aforementioned seven quasigroup properties.

From Table 2 we can see that symmetry breaking technique is highly efficient. *CLS(6)-Ort* and *CLS(8)-Ort* can only be solved with the symmetry breaking technique. From Table 3 we can see that transversal matrix significantly improves the solving efficiency. For almost all problems, the algorithm with transversal matrix is faster than the one without it.

At last we show the number of variables and the number of clauses (in the best method) of each problem in Table 4.

Table 4. The number of variables and clauses in each case

	Vars	Clauses		Vars	Clauses		Vars	Clauses
CLS(6)-Odr	432	73830	CLS(8)-Odr	1024	628360	CLS(10)-Odr	2000	3245210
CLS(6)-Ide	432	73830	CLS(8)-Ide	1024	628360	CLS(10)-Ide	2000	3245210
CLS(6)-Q1-7	432	75120	CLS(8)-Q1-7	1024	622448	CLS(10)-Q1-7	2000	3255200
CLS(6)-Ort	864	186540	CLS(8)-Ort	2048	1486096	CLS(10)-Ort	4000	7390420

6 Conclusion

This paper describes an application of SAT solvers to an important combinatorial structures: Costas Latin Squares. The existence of Costas Latin Squares with specified properties are difficult for mathematical methods. We present two effective solving strategies for these problems: symmetry breaking and transversal matrix. As a result, we find some new instances and prove the non-existence of a number of cases for even order $n \leq 10$. In the future, we will investigate more challenging cases, such as the orthogonal Costas Latin Squares of order 10, as well as Costas Latin Squares of order $n \geq 12$. We believe that finding Costas Latin Squares can be an interesting benchmark for SAT solvers.

Acknowledgments. This work has been supported by the National Natural Science Foundation of China (NSFC) under grant No.61972384, and the Key Research Program of Frontier Sciences, Chinese Academy of Sciences under grant number QYZDJ-SSW-JSC036. Feifei Ma is also supported by the Youth Innovation Promotion Association CAS under grant No. Y202034. We thank professor Lie Zhu at SooChow university for suggesting these open problems and his valuable advice. We also thank the anonymous reviewers for their comments and suggestions.

References

1. Bright, C., Ganesh, V., Heinle, A., Kotsireas, I., Nejati, S., Czarnecki, K.: MathCheck2: A SAT+CAS verifier for combinatorial conjectures. In: Gerdt, V.P., Koepf, W., Seiler, W.M., Vorozhtsov, E.V. (eds.) CASC 2016. LNCS, vol. 9890, pp. 117–133. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45641-6_9
2. Costas, J.P.: A study of a class of detection waveforms having nearly ideal range-doppler ambiguity properties. Proc. IEEE **72**(8), 996–1009 (1984)
3. Dinitz, J., Ostergard, P., Stinson, D.: Packing costas arrays. J. Comb. Math. Comb. Comput. **80**, 02 (2011)
4. Etzion, T.: Combinatorial designs with costas arrays properties. Discrete Math. **93**(2–3), 143–154 (1991)
5. Heule, M.J.H.: Schur number five. In: McIlraith, S.A., Weinberger, K.Q. (eds.) 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, 2–7 February 2018, pp. 6598–6606. AAAI Press (2018)
6. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving and verifying the Boolean Pythagorean triples problem via cube-and-conquer. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 228–245. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_15
7. Huang, P., Liu, M., Ge, C., Ma, F., Zhang, J.: Investigating the existence of orthogonal golf designs via satisfiability testing. In: Davenport, J.H., Wang, D., Kauers, M., Bradford, R.J., (eds.) Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation, ISSAC 2019, Beijing, China, 15–18 July 2019, pp. 203–210. ACM (2019)

8. Huang, P., Ma, F., Ge, C., Zhang, J., Zhang, H.: Investigating the existence of large sets of idempotent quasigroups via satisfiability testing. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS (LNAI), vol. 10900, pp. 354–369. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94205-6_24
9. Knuth, D.E.: The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 2. Pearson Education India, Noida
10. Ma, F., Zhang, J.: Finding orthogonal latin squares using finite model searching tools. Sci. China Inf. Sci. **56**(3), 1–9 (2013)
11. Zhang, H.: Combinatorial designs by SAT solvers. In: Biere, A., Heule, M., van Maaren, H., Walsh, T., (eds.) Handbook of Satisfiability, volume 185 of Frontiers in Artificial Intelligence and Applications, pp. 533–568. IOS Press (2009)
12. Zulkoski, E., Bright, C., Heinle, A., Kotsireas, I.S., Czarnecki, K., Ganesh, V.: Combining SAT solvers with computer algebra systems to verify combinatorial conjectures. J. Autom. Reason. **58**(3), 313–339 (2017)