Aalto University

OPEN ACCESS

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Kokkala, Janne I.; Östergård, Patric R.J.

Kirkman triple systems with subsystems

Published in:
Discrete Mathematics

DOI:
10.1016/j.disc.2020.111960

Published: 01/09/2020

Document Version
Peer reviewed version

Published under the following license:
CC BY-NC-ND

Please cite the original version:
Kokkala, J. I., & Östergård, P. R. J. (2020). Kirkman triple systems with subsystems. *Discrete Mathematics*, *343*(9), [111960]. https://doi.org/10.1016/j.disc.2020.111960

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

Kirkman Triple Systems with Subsystems*

Janne I. Kokkala[1,2] and Patric R. J. Östergård[1]
[1]Department of Communications and Networking
Aalto University School of Electrical Engineering
P.O. Box 15400, 00076 Aalto, Finland
[2] School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
SE-100 44, Stockholm, Sweden

Abstract

A Steiner triple system of order v, STS(v), together with a resolution of its blocks is called a Kirkman triple system of order v, KTS(v). A KTS(v) exists if and only if $v \equiv 3 \pmod 6$. The smallest order for which the KTS(v) have not been classified is $v = 21$, which is also the smallest order for which the existence of a doubly resolvable STS(v) is open. Here, KTS(21) with STS(7) and STS(9) subsystems are classified, leading to more than 13 million KTS(21). In this process, systems missing from an earlier classification of KTS(21) with nontrivial automorphisms are encountered, so such a classification is redone.

Keywords: automorphism group, doubly resolvable, Kirkman triple system, resolution, Steiner triple system, subsystem.

1 Introduction

A *Steiner triple system* of order v, briefly STS(v), is a collection of 3-element subsets, called *blocks*, of a v-set of *points*, such that every pair of points occurs

*Supported in part by the Academy of Finland, Grant No. 289002. The first author was also supported in part by the Swedish Research Council grant 2016-00782. The first author is currently at Faculty of Engineering, LTH, Lund University, Sweden and Department of Computer Science, University of Copenhagen, Denmark

in exactly one block. Each point occurs in exactly $r = (v-1)/2$ blocks of an STS(v), and such a triple system has exactly $b = v(v-1)/6$ blocks. These values must obviously be integers, which gives necessary conditions for the existence of an STS(v). It is well known that these conditions are also sufficient.

Theorem 1. *For $v \geq 3$, an* STS(v) *exists if and only if either $v \equiv 1 \pmod 6$ or $v \equiv 3 \pmod 6$.*

A partition of all points into blocks of an STS(v) is called a *parallel class*. A necessary condition for such partitions to exist is then that $v \equiv 3 \pmod 6$. Moreover, a partition of all blocks of an STS(v) into parallel classes is a *resolution*, and an STS(v) that has at least one resolution is said to be *resolvable*. An STS(v) together with a resolution of its blocks is a *Kirkman triple system*, briefly KTS(v). The basic necessary condition is sufficient also here.

Theorem 2. *For $v \geq 3$, a* KTS(v) *exists if and only if $v \equiv 3 \pmod 6$.*

For more information about these results and various other aspects of triple systems, the reader is referred to [5].

An STS(v) that has two resolutions with the property that every pair of parallel classes, one from each of the resolutions, intersects in at most one block is said to be *doubly resolvable*. The following result is from [4].

Theorem 3. *For $v \geq 3$, a doubly resolvable* STS(v) *exists if and only if $v \equiv 3 \pmod 6$ with the exception of $v \in \{9, 15\}$ and a finite number of possible further exceptions.*

The number of possible exceptions in Theorem 3 are reduced to 12 in [2], the two smallest being $v = 21$ and $v = 141$.

Two STS(v) are *isomorphic* if there exists a bijection between the point sets that maps blocks onto blocks; such a bijection is an *isomorphism*. An *automorphism* of a triple system is an isomorphism of the triple system onto itself. The set of all automorphisms forms the *automorphism group* of the triple system under composition. Two KTS(v) are *isomorphic* if there exists a bijection between the point sets that maps parallel classes onto parallel classes. The other related concepts are defined for KTS(v) in an analogous way as for STS(v). The number of structures refers to the number of isomorphism classes in the sequel, unless otherwise mentioned.

There is trivially a unique KTS(3), and the unique KTS(9) comes from the four sets of mutually parallel lines of the affine plane of order 3. There are four resolvable STS(15) and seven KTS(15), a result obtained already in

1917 in Pieter Mulder's PhD thesis [16]; see also [6]. The next open case is KTS(21), which is also of interest as the smallest open case for the existence of doubly resolvable STS(v).

Despite several and serious attempts during the last decades, by the authors and others, the KTS(21) have not yet been classified. One reason for this is that a combinatorial explosion of the number of structures sets in at this point, similarly to the behavior of the number of Steiner triple systems in the step from STS(15) to STS(19) [9].

A common approach for combinatorial structures that are difficult to classify is to restrict the classification to a subset of structures with some additional properties. For example, a classification of KTS(21) with nontrivial automorphisms is presented in [3]. In the current work, we consider KTS(21) with proper nontrivial STS(w) subsystems. Possible such subsystems are STS(7) and STS(9).

The paper is organized as follows. The algorithm for classifying KTS(21) with STS(7) and STS(9) subsystems and the computational results are presented in Section 2; the numbers of such structures are 12520021 and 668553, respectively, with 19935 KTS(21) in the intersection of the sets. Hence there are more than 13 million KTS(21). When comparing the structures with those obtained in [3], it turns out that some systems are missing in [3]. The classification of KTS(21) with nontrivial automorphisms is therefore redone, and the results are presented in Section 3.

2 Classification

2.1 Kirkman Triple Systems as Codes

A q-ary *code* of length n is a subset of $\mathcal{A}^n$, where $\mathcal{A}$ is an alphabet of size q. Such a code with size M and minimum (Hamming) distance d is called an $(n, M, d)_q$ code. In this work, we use $\{0, 1, \ldots, q-1\}$ as the alphabet.

Two codes are *equivalent* if one can be obtained from the other by permuting coordinates and permuting the symbols in each coordinate separately. Such operations that map a code C onto itself are *automorphisms* of C. The set of all automorphisms forms the *automorphism group* of C under composition, denoted by $\mathrm{Aut}(C)$.

A code is *equidistant* if the Hamming distance between each pair of distinct codewords is the same. An $(n, M, d)_q$ code is *equireplicate* if in each coordinate, each symbol occurs in M/q codewords. An $(n, M, d)_q$ is further called *optimal* if it has the largest possible M for the given values of n, d, and q. The following result from [21] plays a central role here.

Theorem 4. *If q divides M and the parameters of an $(n, M, d)_q$ code satisfy*

$$\binom{M}{2} d = n \binom{q}{2} (M/q)^2,$$

then the code is equidistant, equireplicate, and optimal.

The codes characterized by Theorem 4 are often called *optimal equidistant* (OE) codes. There is a close connection between OE codes and resolvable designs; for a general discussion, see [10, Sect. 2.3.2]. We will here give some details for the KTS(21), to which we can associate $(10, 21, 9)_7$ OE codes in the following way.

We label the parallel classes of a KTS(21) with $1, 2, \ldots, 10$ and the blocks in every parallel class with $0, 1, \ldots, 6$. Now, for each point, we get a codeword $c = (c_1, \ldots, c_{10})$ by letting c_i be the label of the block of parallel class i in which the point occurs. It is not difficult to see the connection between the parameters and properties of the designs and the codes. Moreover, two KTS(21) are isomorphic if and only if the corresponding $(10, 21, 9)_7$ OE codes are equivalent.

An STS(w) subsystem of a KTS(21) corresponds to a subcode of size w of the corresponding OE code with the property that, for every coordinate, each symbol occurs either 0, 1, or 3 times. Therefore, the problem of classifying KTS(21) with STS(w) subsystems is equivalent to the problem of classifying $(10, 21, 9)_7$ OE codes that contain such subcodes. We use the framework of codes in the algorithm to be presented now.

2.2 Main Algorithm

Our main approach is a backtrack search that recursively adds one codeword at a time.

2.2.1 Subsystems as Codes

As we consider Kirkman triple systems with proper nontrivial $STS(w)$ subsystems, we may utilize the following result [5, Lemma 6.1].

Theorem 5. *An* STS(v) *can have a proper sub-*STS(w) *only if* $w < v/2$.

Here $v = 21$ and possible proper nontrivial subsystems are STS(7) and STS(9). The starting points of the search are then designs with these parameters and all possible ways of placing the blocks into 10 sets in a non-overlapping way, up to isomorphism (and all this in the coding framework).

Since the blocks of an STS(7) intersect pairwise, those blocks must be in different parallel classes. Hence, the substructure of a Kirkman triple system in the seven points of an sub-STS(7) is unique (the subsystem is shown in bold type):

0000000000
0111**1**11111
02222**2**2222
1**0**3**1**3**2**3333
2**0**43**1**3**2**444
33**01**44**2**555
44**0**4**1**2**4**666

For a KTS(21) with a sub-STS(9), the set of 12 blocks of the subsystem must intersect all 10 parallel classes. Namely, a parallel class without any block of the subsystem would have nine points from different blocks, which is not possible since there are seven blocks in a parallel class. A similar argument gives the well-known necessary condition $v \geq 3w$ for a KTS(v) to have a sub-KTS(w); see, for example, [18].

There are two possibilities for the distribution of the blocks of a sub-STS(9) amongst the parallel classes: $3+1+1+\cdots+1$ and $2+2+1+1+\cdots+1$. Up to symmetry, there is just one possible structure in each of these cases:

0000000000	**0000**000000
0111**11**1111	**0**111**1**11111
0222222**22**	**0**22222**22**2
1033133233	**10**33132333
1304314324	**110**4343244
1440441442	**1**340414425
2055551525	20555152**56**
2506165652	3406156526
2660616266	4160662666

2.2.2 Augmenting Partial Codes

Given a partial code C of the current search, a word c is said to be *admissible* if it is at Hamming distance 9 from each codeword in C and there is no coordinate in which a symbol occurs more than three times in $C \cup \{c\}$.

For a partial code C, let $G(C)$ be the graph that has all admissible words as vertices and an edge between each pair of admissible words that have Hamming distance exactly 9. Now $C \cup C'$ (for C' disjoint from C) is a $(10, 21, 9)_7$ OE code if and only if C' corresponds to a clique of size $21 - |C|$ in

$G(C)$. Note that the condition on the number of occurrences of a coordinate–symbol pair is redundant when considering final codes, since by Theorem 4 equidistance is sufficient to guarantee that the final code is equireplicate. However, the condition is useful for limiting the number of candidate words considered at each node, which makes a tailored backtrack search perform much better than a general clique-finding algorithm.

When augmenting a partial code C, in each step we (heuristically) choose a coordinate–symbol pair (i, v) and find all admissible words that have symbol v in coordinate i, and call the search recursively for each code obtained by adding such a codeword to C. The selection of the pair (i, v) is here done as follows and depends on whether the code C is one of the initial codes (in Section 2.2.1) or not: amongst the pairs (i, v) that occur exactly once (resp. twice) in C, the one that gives the smallest candidate set of new codewords is chosen. In the search, there is a tradeoff between time used for computing the possible candidate set sizes and the potential gain of doing it exactly, so in our implementation the actual candidate set used for the selection may contain some inadmissible words; see Section 2.3.1 for details.

Remark 1. *Except for the initial codes corresponding to a sub-*STS(7) *and a sub-*STS(9)*, there exist pairs* (i, v) *that are fulfilled by two codewords whenever the size of the code is smaller than* 21*. Namely, a lack of such pairs would mean that an* STS(w) *has been encountered, and the proper nontrivial subsystems of a* KTS(21) *are* STS(7) *and* STS(9)*, and an* STS(9) *cannot have a sub-*STS(7) *by Theorem* 5*.*

Isomorph rejection of final and partial structures is an important ingredient in any classification algorithm. There is here one straightforward way of carrying out partial isomorph rejection: given a partial code C, adding candidate words c and c' that differ only in coordinates where they have a symbol that does not occur in C in that coordinate leads to two equivalent codes. Throughout the search, we therefore consider only new words that in each coordinate have either a symbol that already appears in that coordinate in the code or the smallest previously unused symbol.

We carry out full isomorph rejection on the first few levels of the search—for code size up to 10 or 11 (when starting from sub-STS(7) and sub-STS(9), respectively)—and on the final level when the code is complete. The most time-consuming part of the search is the phase starting from level 10 or 11 onwards when the full isomorph rejection is not carried out and we simply add admissible words in an exhaustive manner (up to the symmetry of previously unused symbols in each coordinate). We call this phase the *main phase* and the initial codes of the main phase *seeds*. Consequently, this is the crucial part what comes to efficient implementation. Some details on the

implementation of the main phase are given in Section 2.3.1, and isomorph rejection is discussed in Section 2.3.2.

2.3 Implementation Details

2.3.1 Data Structure for Candidate Words

Generating the set of admissible codewords from scratch at each node of the search tree separately would become the bottleneck in the main phase. Instead, we maintain a list of candidate words incrementally during the search. In the following, we describe implementation details of the main phase of the search starting from a seed C.

We first precompute the graph $G(C)$ by computing the set S of all admissible words and the neighbourhood of each word $c \in S$, that is, set S_c of all words in S that have Hamming distance 9 to c. Further, we also generate the subset $S_{(i,v)}$ of all words in S that contain that pair, for each (i, v) separately. When the main phase starts, the number of admissible words is small enough (under 7000) for the subsets of S to be efficiently implemented as bitmasks of length $|S|$. Bit operations can then be applied during the search to efficiently compute intersections of these subsets.

A node of the search corresponds to a partial code $C \cup C'$, where C' contains the words that have been added in parent nodes. All admissible words are contained in the set $\bigcap_{c \in C'} S_c$, which is maintained incrementally during the search. For the coordinate–symbol pair selection heuristic, we compute the set $S_{(i,v)} \cap \bigcap_{c \in C'} S_c$ for all (i, v), and select the smallest of these to be the candidate set. Finally, when iterating over the candidate set, we reject words that do not satisfy the condition of having at most 3 occurrences of each coordinate–symbol pair in the resulting partial code or the condition that in each coordinate there is either a symbol that already appears in that coordinate in the code or the smallest previously unused symbol.

In addition, since the time used for bit operations is linear in the size of a bitmask, the precomputation is redone whenever the set of candidate words falls below a certain limit (in preliminary testing, the optimal choice seemed to be 2000). Further, when iterating over the candidate set described by a bitmask, checking all individual bits would take a significant portion of time since in most nodes the proportion of true bits in the bitmask is small. Therefore some performance improvement can be obtained by precomputing the list of all indices corresponding to elements in $S_{(i,v)}$ for each (i, v) and checking only those.

2.3.2 Equivalence of Codes

The computational problem of detecting equivalence of codes can be reduced to the graph isomorphism problem in the following standard way [20]. To an $(n, M, d)_q$ code, we associate the following colored graph with $nq + M$ vertices. The graph contains n copies of the complete graph of order q, colored with the first color, where each copy corresponds to a coordinate and each vertex corresponds to a symbol in that coordinate. In addition, the graph contains M vertices, colored with the second color, one for each codeword. For each codeword c and each coordinate i, there is an edge from the vertex corresponding to c to the vertex corresponding to the coordinate–symbol pair (i, c_i). Now two codes are equivalent if and only if the corresponding graphs are isomorphic, and the automorphism group of the graph is isomorphic to the automorphism group of the code. We use the graph isomorphism library *nauty* [15] in our computations.

There are several general approaches for isomorph rejection when classifying combinatorial structures [10, Sect. 4]. The number of objects after isomorph rejection is in our case not prohibitively large, so an arbitrary method can be used for this.

2.4 Results

The classification of KTS(21) with proper nontrivial subsystems was run in a cluster consisting of Intel Xeon E5 family processors. The computation times below refer to a single logical core of such a processor. Starting from the STS(7), generating the codes (seeds) of size 10 took 1682 core-seconds, and classifying the KTS(21) containing a STS(7) took 4386 core-days, that is, about 12 core-years. Starting from the two STS(9), generating the codes (seeds) of size 11 took 49 core-seconds, and classifying the KTS(21) containing a STS(9) took 119 core-hours.

The search encountered 12511221 KTS(21) containing an STS(7) and 668553 KTS(21) containing an STS(9). Of these, 19935 contain both an STS(7) and an STS(9). The numbers of designs are listed in Table 1 according to the order of the automorphism group and in Table 2 according to the number of subsystems.

All these designs (codes) are made available at [14]. Experiments reveal that at least hundreds of thousands of more structures can be obtained from these—and those to be considered in Section 3—using different types of switching (see, for example, [3, Sect. 4]). We do not further discuss these here as they conjecturally form just a tiny fraction of the KTS(21) still undiscovered.

Table 1: KTS(21) with subsystems

|Aut|	Sub-STS(7)	Sub-STS(9)	Both
1	12511221	665959	19531
2	7460	1413	287
3	1272	1161	110
6	43	1	
7	3		
9	15	14	6
21	5	1	1
27		4	
63	2		
Total	12520021	668553	19935

Table 2: Number of subsystems

STS(7)	STS(9)	#
0	1	648255
0	3	359
0	7	4
1	0	12433428
1	1	18728
1	3	61
2	0	54112
2	1	225
2	3	1
3	0	12540
3	1	847
3	3	51
3	7	20
4	0	6
4	1	2

For all new KTS(21) obtained in this work, it was checked whether they are doubly resolvable, but none of them are.

2.5 Double Counting

To validate the computational results, we perform a consistency check by double counting. For $w = 7, 9$ separately, we count in two ways the number of pairs (C, D) where C is a $(10, 21, 9)_7$ OE code and $D \subseteq C$ is a code that corresponds to a STS(w) subsystem.

The first count comes from the final classification results. By the orbit–stabilizer theorem, the equivalence class of a $(10, 21, 9)_7$ OE code C has size $10!(7!)^{10}/|\mathrm{Aut}(C)|$. Let $S_w(C)$ be the number of subsets of C of size w that correspond to a STS(w) subsystem. Now the total count is

$$\sum_C \frac{10!(7!)^{10}}{|\mathrm{Aut}(C)|} S_w(C),$$

where the sum is taken over the codes output by the classification program.

The second count comes from statistics stored during the search. Let $N(D)$ be the number of distinct $(10, 21, 9)_7$ OE codes that contain the $(10, w, 9)_7$ code D from which the search started. This is precisely the number of different codes that would be found starting from D if no isomorph rejection was performed. Now, since we carry out full isomorph rejection during the first couple of levels of the search, to find $N(D)$ we need to keep track of a multiplier, which is updated whenever equivalent subcodes are rejected. Moreover, we need to take into account that the search chooses only one of the ways to permute the symbols unused in D. Finally, we note that precisely in the step of augmenting a code corresponding to the sub-STS(w), there is just one old codeword with the chosen coordinate–symbol pair (i, v), and hence two of the words to be added later will have this pair (cf. Remark 1). Consequently, the algorithm, without any isomorph rejection, would obtain each complete code exactly twice. Since the equivalence class of D has size $10!(7!)^{10}/|\mathrm{Aut}(D)|$, the final count is

$$\sum_D \frac{10!(7!)^{10}}{|\mathrm{Aut}(D)|} N(D),$$

where the sum is taken over all possible D from which the search started (one for $w = 7$, with $|\mathrm{Aut}(D)| = 129024$ and two for $w = 9$ with $|\mathrm{Aut}(D)| = 2592$ and $|\mathrm{Aut}(D)| = 32$ (in the order in which they are displayed in Section 2.2.1).

2.6 Transversal Designs as Subdesigns

Steiner triple systems are arguably the most natural subdesigns to consider in the study of Kirkman and Steiner triple systems. However, there are also other possible subdesigns, such as transversal designs. A TD(k, n) *transversal design* is a design on kn points partitioned into k *groups* of size n and consists of k-element blocks, such that every pair of points occurs in either exactly one block or in exactly one group, but not both.

In Steiner triple systems we have 3-element blocks, so transversal designs that can occur as subdesigns have parameters TD$(3, n)$. Such transversal designs are actually set systems corresponding to latin squares, cf. [1].

In the smallest nontrivial case, TD$(3, 2)$, there is a unique latin square and corresponding set system. This set system, with 4 blocks on 6 points, is a well-known structure known as a *quadrilateral* or *Pasch configuration*. The unique STS(7) contains seven quadrilaterals, so all Kirkman triple systems with STS(7) subsystems contain TD$(3, 2)$ subdesigns. Similarly, since the unique STS(9) contains four TD$(3, 3)$ subdesigns, all Kirkman triple systems with STS(9) subsystems contain TD$(3, 3)$ subdesigns.

In recent work [7], the KTS(21) with TD$(3, 6)$ subdesigns are considered, and it is shown [7, Proposition 5] that such systems must have more than one STS(9) subsystem. Something stronger can actually be proved.

Theorem 6. *A* KTS(21) *has more than one* STS(9) *subsystem if and only if it has a* TD$(3, 6)$ *subdesign.*

Proof. Due to [7, Proposition 5], it remains to show that with more than one STS(9) subsystem, there will be a TD$(3, 6)$ subdesign. This will now be proved more generally for STS(21).

Consider an STS(21) with (at least) two STS(9) subsystems. Further let the point set of the STS(21) be partitioned as $A \cup B \cup C \cup D$, where $A \cup D$ and $B \cup D$ are the point sets of two STS(9) subsystems. Since the intersection of the point sets of two subsystems induces a subsystem, $|D| \in \{0, 1, 3\}$.

Every block that intersects both A and B must intersect C, so there are 36 blocks $\mathcal{B}$ of the form $\{a, b, c\}$, $a \in A$, $b \in B$, $c \in C$. If $|C| < |A| = |B|$, some pair will be covered more than once; this implies that $|D| = 0$ and $|D| = 1$ are not possible and we get that $|A| = |B| = |C| = 6$ and $|D| = 3$. The blocks $\mathcal{B}$ then form a TD$(3, 6)$. (One may further show that the point set $C \cup D$ induces a third STS(9) subsystem.) □

By Table 2 and Theorem 6, there are 496 KTS(21) with TD$(3, 6)$ subdesigns. The number of resolvable STS(21) that have TD$(3, 6)$ subdesigns, 393, is reported already in [7].

Finally, the KTS(21) with TD$(3,7)$ subdesigns are the KTS(21) with three disjoint STS(7) subsystems. These systems are classified in [11]. There are 11466 resolvable STS(21) with TD$(3,7)$ subdesigns and these have 13036 resolutions, giving the number of KTS(21) with TD$(3,7)$ subdesigns. Investigation of the systems classified here corroborate these numbers.

3 KTS(21) with Nontrivial Automorphisms

A tailored algorithm for classifying the KTS(21) with nontrivial automorphisms is presented in [3]. Since a classification of the STS(21) with nontrivial automorphisms is currently available [8], the KTS(21) with nontrivial automorphisms can now be classified in a rather direct way from those Steiner triple systems. Namely, since an automorphism of a Kirkman triple system maps parallel classes onto parallel classes, it maps blocks onto blocks, and the automorphism group of a Kirkman triple system is therefore a subgroup of the automorphism group of the underlying Steiner triple system. For order 21, the computing time is not a bottleneck when applying either the original tailored algorithm or the approach used here.

One way of using the STS(21) to classify the KTS(21) is to apply the following algorithm to each Steiner triple system. First construct all possible parallel classes, and then partition the blocks of the system into parallel classes, in all possible ways. Finally, isomorph rejection has to be carried out. For this approach and variants of it, see [19] and [10, Sect. 6.3.1]. Let us here elaborate on some of the details.

In Kirkman triple systems, no blocks occur more than once, which simplifies the approach. Both the stage of constructing parallel classes and of forming resolutions can be considered as instances of the exact cover problem. The (decision version of the) *exact cover problem* asks whether a set S can be partitioned into subsets, where the candidate subsets are listed in a collection $\mathcal{S}$. Instances of the exact cover problem are recurrent in constructive combinatorics, and related algorithms are of great interest [13]. If all sets in $\mathcal{S}$ have the same size s, then one may also use the (suboptimal) approach of finding cliques of size $|S|/s$ in a graph with one vertex for each element of $\mathcal{S}$ and edges whenever the corresponding sets are disjoint. Two available computational tools for the exact cover problem and the clique problem are libexact [12] and Cliquer [17], respectively.

Starting from the STS(21) with nontrivial automorphisms, the KTS(21) with nontrivial automorphisms were classified using the approach described above in 8.3 core-hours, and the same results were obtained in another, independent implementation. The result is presented in Table 3, and the classified

structures can be obtained from [14]. The STS(21) with nontrivial automorphisms further underlie 1992 KTS(21) with no nontrivial automorphisms.

Table 3: KTS(21) with nontrivial automorphisms

|Aut|	[3]	Here
2	49835	49835
3	13035	16806
5	48	48
6	189	189
7	3	3
9	45	45
21	5	5
27	4	4
63	2	2
Total	63166	66937

Due to the discrepancy between the results obtained here and in [3], we recalculate other affected results in [3]. In Table 4, which corresponds to [3, Table 1], the number of KTS(21) with automorphisms of order 3 in different classes is given. A KTS(21) with an automorphism of order 3 that fixes e elements, c parallel classes, and b blocks is said to be in class (e, c, b). The entries differing from those in [3] are here and in subsequent tables shown in italics.

Table 4: Partitions into classes

Class||Aut|	3	6	9	21	27	63	Total
(0,1,1)	1004	17	11			1	1033
(0,1,4)	20	2	9			1	32
(0,1,7)		1	4				5
(0,4,4)	*1387*	34	20	1	4	1	*1447*
(0,4,7)	20		23			1	44
(0,7,7)	55	4	24			2	85
(3,1,1)	*11657*	100	*19*		*4*		*11780*
(3,1,4)	*2647*	31	20	4	4	2	*2708*
(3,1,7)	16		16		4		36

Table 5 corresponds to [3, Table 5] and shows how the numbers of isomor-

phism classes of KTS(21) with an automorphism of order 3 are distributed amongst the underlying STS(21).

Table 5: Underlying STS(21)

#KTS	#STS	#KTS	#STS
1	*12698*	6	*6*
2	*1751*	7	2
3	*130*	8	4
4	*82*	10	1
5	*5*	16	1

Table 6 corresponds to [3, Table 6] and gives the orders of the automorphism groups of the STS(21) underlying the KTS(21) with an automorphism of order 3.

Table 6: Group orders of underlying STS(21)

|Aut|	#STS	|Aut|	#STS
3	*14536*	42	4
6	*113*	72	1
9	11	126	1
18	5	294	1
21	1	882	1
24	4	1008	1
27	1		

The last table, Table 7, which corresponds to [3, Table 4], categorizes the KTS(21) with an automorphism of order 3 based on the number of orthogonal parallel classes (#OPC). This information is interesting in the study of doubly resolvable designs. There are indeed 11 KTS(21) with an automorphism of order 3 with the property that every block belongs to at least one parallel class that is orthogonal to all ten parallel classes of the KTS(21), as stated in [3]. Consequently, no new candidates for doubly resolvable STS(21) emerge in this part of the current work.

Finally, six of the KTS(21) with an automorphism of order 3 contains no quadrilateral (Pasch configuration), not four as stated in [3].

Table 7: Orthogonal parallel classes

#OPC	#KTS	#OPC	#KTS	#OPC	#KTS
0	*5694*	9	*49*	18	3
1	*4970*	10	*42*	20	8
2	*1761*	11	*48*	21	2
3	*1881*	12	4	23	7
4	*1362*	13	*8*	25	2
5	*521*	14	12	26	2
6	*321*	15	5	27	1
7	*202*	16	1	32	7
8	*130*	17	7	53	1

Acknowledgments

The authors thank Charlie Colbourn and Petteri Kaski for providing electronic data and for helpful discussions. The authors also thank a referee for suggesting the inclusion of transversal subdesigns.

References

[1] R. J. R. Abel, C. J. Colbourn, and J. H. Dinitz, Mutually orthogonal latin squares (MOLS), in C. J. Colbourn and J. H. Dinitz (Eds.), *Handbook of Combinatorial Designs*, 2nd ed., Chapman & Hall/CRC, Boca Raton, 2007, pp. 160–193.

[2] R. J. R. Abel, E. R. Lamken, and J. Wang, A few more Kirkman squares and doubly near resolvable BIBDs with block size 3, *Discrete Math.* **308** (2008), 1102–1123.

[3] M. B. Cohen, C. J. Colbourn, L. A. Ives, and A. C. H. Ling, Kirkman triple systems of order 21 with nontrivial automorphism group, *Math. Comp.* **71** (2002), 873–881.

[4] C. J. Colbourn, E. R. Lamken, A. C. H. Ling, and W. H. Mills, The existence of Kirkman squares—doubly resolvable $(v, 3, 1)$-BIBDs, *Des. Codes Cryptogr.* **26** (2002), 169–196.

[5] C. J. Colbourn and A. Rosa, *Triple Systems*, Clarendon Press, Oxford, 1999.

[6] F. N. Cole, Kirkman parades, *Bull. Amer. Math. Soc.* **28** (1922), 435–437.

[7] Y. Guan, M. Shi, D. S. Krotov. The Steiner triple systems of order 21 with a transversal subdesign TD$(3, 6)$, *Probl. Inf. Transm.*, to appear.

[8] P. Kaski, Isomorph-free exhaustive generation of designs with prescribed groups of automorphisms, *SIAM J. Discrete Math.* **19** (2005), 664–690.

[9] P. Kaski and P. R. J. Östergård, The Steiner triple systems of order 19, *Math. Comp.* **73** (2004), 2075–2092.

[10] P. Kaski and P. R. J. Östergård, Classification Algorithms for Codes and Designs, Springer, Berlin, 2006.

[11] P. Kaski, P. R. J. Östergård, S. Topalova, and R. Zlatarski, Steiner triple systems of order 19 and 21 with subsystems of order 7, *Discrete Math.* **308** (2008), 2732–2741.

[12] P. Kaski and O. Pottonen, libexact user's guide, version 1.0, HIIT Technical Reports 2008-1, Helsinki Institute for Information Technology HIIT, 2008.

[13] D. E. Knuth, Dancing links, in: J. Davies, B. Roscoe, and J. Woodcock (Eds.), *Millennial Perspectives in Computer Science*, Palgrave, Houndmills, 2000, pp. 187–214.

[14] J. I. Kokkala and P. R. J. Östergård, Dataset for Kirkman triple systems with subsystems [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.3699565 (Aug 29, 2019).

[15] B. D. McKay and A. Piperno, Practical graph isomorphism, II, *J. Symbolic Comput.* **60** (2014), 94–112.

[16] P. Mulder, *Kirkman-systemen*, PhD Thesis, Rijksuniversiteit Groningen, 1917.

[17] S. Niskanen and P. R. J. Östergård, Cliquer user's guide, version 1.0, Technical Report T48, Communications Laboratory, Helsinki University of Technology, Espoo, 2003.

[18] R. Rees and D. R. Stinson, On the existence of Kirkman triple systems containing Kirkman subsystems, *Ars Combin.* **26** (1988), 3–16.

[19] P. R. J. Östergård, Constructing combinatorial objects via cliques, in B. S. Webb (Ed.), *Surveys in Combinatorics, 2005*, Cambridge University Press, Cambridge, 2005, pp. 57–82.

[20] P. R. J. Östergård, T. Baicheva, and E. Kolev, Optimal binary one-error-correcting codes of length 10 have 72 codewords, *IEEE Trans. Inform. Theory* **45** (1999), 1229–1231.

[21] N. V. Semakov and V. A. Zinovev, Equidistant q-ary codes with maximal distance and resolvable balanced incomplete block designs (in Russian), *Problemy Peredachi Informatsii* **4** (1968), 3–10. English translation in *Problems Inform. Transmission* **4** (1968), 1–7.