Classification of simple 2-$(11, 3, 3)$ designs

G. B. Khosrovshahi[a,b] B. Tayfeh-Rezaie[a,1]

[a]*School of Mathematics,*
IPM (Institute for studies in theoretical Physics and Mathematics),
P.O. Box 19395-5746*, Tehran, Iran*

[b]*Center of Excellence of Biomathematics, University of Tehran, Tehran, Iran*

Abstract

We present an orderly algorithm for classifying triple systems. Subsequently, we show that there exist exactly 7,038,699,746 nonisomorphic simple 2-$(11, 3, 3)$ designs. The method is also used to confirm the previously accomplished classifications of 2-(8,3,6), 2-$(12, 3, 2)$ and 2-$(19, 3, 1)$ designs.

Keywords: Block designs, triple systems, orderly algorithms, automorphism groups
MR Subject Classification: 05B05

1 Introduction

A 2-(v, k, λ) *design* is a pair $\mathcal{D} = (X, B)$ where X is a v-set of *points* and B is a collection of k-subsets of X (called *blocks*) such that any pair of distinct points is contained in exactly λ blocks. If no blocks are identical, then $\mathcal{D}$ is called *simple*. The trivial necessary conditions for the existence of $\mathcal{D}$ are $k-1|\lambda(v-1)$ and $k(k-1)|\lambda v(v-1)$. For given v and k, the smallest value of λ satisfying these conditions is denoted by $\lambda_{\min} = \lambda_{\min}(v, k)$. Let σ be a permutation on X. By $\sigma\mathcal{D}$, an *isomorphic copy* of $\mathcal{D}$, we refer to the design $(X, \sigma B)$ in which the collection σB is obtained from B by applying σ (Note that σ induces a permutation $\bar{\sigma}$ on k-subsets of X. For the sake of simplicity, we have written σB instead of $\bar{\sigma} B$.). An *automorphism* of $\mathcal{D}$ is a permutation σ such that $\sigma\mathcal{D} = \mathcal{D}$ and the group of all automorphisms of $\mathcal{D}$ is called the *automorphism group* of $\mathcal{D}$.

One of the central issues of combinatorial design theory is the classification of basically different types in a given class. For historical, statistical and many other reasons, 2-designs have drawn more attention among combinatorial design theorists and consequently their classifications

[1]`Corresponding author, email: tayfeh-r@ipm.ir`

have been pursued by some specialists of the field. However unfortunately, except for small cases, the problem, if it is at all feasible, has to be carried out by computer. During the last decades some computer algorithms for constructive enumeration of various combinatorial objects were developed. These improved algorithms along with increasing speed of computers have resulted in new classifications in recent years. For a comprehensive survey of the subject, we refer the reader to the recently published monograph [16].

In this paper, we are concerned with the classification problem of 2-$(v, 3, \lambda)$ designs also known as *triple systems*. It is interesting to note that even for some of the small values of v, the classification of 2-$(v, 3, \lambda_{\min})$ designs is still an open problem. Table 1 indicates the state of the art in the classification of these objects. As it is seen from the table, $v = 11$ is the first unknown case waiting for an enumeration. We present an orderly algorithm for the classification of triple systems and subsequently we employ it to classify simple 2-$(11, 3, 3)$ designs. It turns out that there is a total of 7,038,699,746 nonisomorphic simple 2-$(11, 3, 3)$ designs. The computation time was about two days on a cluster of 14 Pentium 4 CPU 2.6 GHz. We also used our program to generate designs with repeated blocks. The program produced more than 2×10^{11} designs after running for about two months on the above mentioned cluster.

Finally, we use the method to confirm the previously known classifications of 2-(8,3,6), 2-$(12, 3, 2)$ and 2-$(19, 3, 1)$ designs. The results coincide with those reported in [22], [25] and [17], respectively.

Table 1 The known results on 2-$(v, 3, \lambda_{\min})$ designs

v	$\lambda_{\min}$	#Designs	References
6	2	1	folklore
7	1	1	folklore
8	6	3,077,244	[22]
9	1	1	folklore
10	2	960	[1, 7, 15]
12	2	242,995,846	[25]
13	1	2	[3, 26]
15	1	80	[4, 11, 21, 30]
19	1	11,084,874,829	[17]

2 The algorithm

The problem of *isomorph-free exhaustive generation* of combinatorial objects is one of the central topics in combinatorics. For a specific family of objects with given properties, the objective is to generate a representative for each of the isomorphism classes of those objects. Any algorithm for an isomorph-free exhaustive generation in general consists of two essential parts. One part deals with the construction of objects and the second part is to reject isomorphic copies of objects during the construction process. These parts are usually performed parallel to each other and have some interactions. For the construction phase, the most natural and widely used method is backtracking which has quite an old history, see for example [9, 29]. The method in its general form can be found in many textbooks among them is [16]. For the isomorph rejection, there are apparently three general schemes in the literature. These are *orderly generation* which was independently introduced by Read [27] and Faradžev [5], *canonical augmentation* which has been proposed by McKay [23] and the *method of homomorphisms* which has been developed by Laue and others [10]. For a thorough discussion of these methods, we refer the reader to [16]. The orderly generation method (also called the Read-Faradžev scheme) has widely been used in enumerations and classifications of combinatorial objects (see [8, 16]). Algorithms based on this scheme are called *orderly* algorithms.

In this section we present an orderly algorithm for the classification of 2-$(v, 3, \lambda)$ designs. In the construction phase, we will use a backtracking algorithm already developed for simple t-designs in [18, 24]. For the isomorph rejection part which is based on the Read-Faradžev scheme, we make use of the notion of canonical forms. A new method is introduced to test the canonicity of subobjects generated during the construction process.

First we briefly explain our backtracking method for the construction of designs. Throughout this section, we let $X = \{1, 2, \ldots, v\}$. Let W^v be a $(0, 1)$-matrix whose rows and columns are indexed by the 2-subsets and 3-subsets of X, respectively, and for a 2-subset T and a 3-subset K, $W^v(T, K) = 1$ if and only if $T \subseteq K$. Let $\mathcal{D} = (X, B)$ be a 2-$(v, 3, \lambda)$ design. We represent $\mathcal{D}$ with a column vector D of dimension $\binom{v}{3}$ whose coordinates are indexed by the 3-subsets of X (ordered lexicographically) and the ith entry is the number of occurrences of the corresponding 3-subset in $\mathcal{D}$. It follows that

$$W^v D = \lambda J,$$

where J is the all-one column vector. We use the above equation to find 2-$(v, 3, \lambda)$ designs. This equation is solved by a backtracking algorithm equipped with the so called *preset* technique developed in [18, 24]. In these references the method is presented for simple designs, however it is straightforward to adapt it for designs with repeated block. The main change is made to the presetting procedure which turns out to be somewhat different for the repeated case. Let S be the set of entries of D corresponding to $v - 2$ nonzero entries of a typical row r of W^v.

In simple designs, exactly λ elements of S must take the nonzero value 1. Hence, in any stage of the backtracking algorithm if it is found that exactly λ $(v-2-\lambda)$ elements of S have been set the value 1 (0), then the remaining elements of S should be preset to 0 (1). In the repeated case, the presetting procedure is only due to nonzero values: If, in any stage of the backtracking algorithm, it is found that the sum of nonzero values of elements of S is equal to λ, then the remaining elements of S should be preset to 0. We refer the reader to [18, 24] for details and here we only note that this backtracking algorithm is in fact a generalization of the *exact cover* algorithm for finding t-designs. The reader can consult [19] for a thorough discussion of the exact cover algorithm and its implementation. We also note that the ideas in this algorithm can be traced at least to [9]. The exact cover algorithm has also been used in the classification of Steiner triple systems of order 19 [17].

For the isomorph rejection part of the algorithm, we make use of the notion of canonical forms. We consider the natural lexicographical ordering between column vectors which is defined by comparing corresponding entries of vectors from top to down. As described above, we represent a 2-$(v,3,\lambda)$ design $\mathcal{D}$ with column vector D whose coordinates are indexed by the 3-subsets of X in the lexicographical order. Therefore, σD is an isomorphic copy of D for any $\sigma \in S_X$, the symmetric group on X. We say that the column vector D is in *canonical form* if $D \geq \sigma D$ for all $\sigma \in S_X$. From the definition, it is clear that in any isomorphism class of designs, exactly one design is in the canonical form. The task of the isomorph rejection part of algorithm is to recognize those partially constructed vectors which do not lead to canonical forms. This recognition in its ultimate form could be quite expensive. Hence, we have to find a fast method to check the canonicity of partial vectors. For this purpose, we make use of the *neighborhood* graphs of 2-$(v,3,\lambda)$ design $\mathcal{D} = (X, B)$. For any $x \in X$, the neighborhood graph associated to x is $G_x = (X \setminus \{x\}, E)$, where $E = \{\{y,z\} : \{x,y,z\} \in B\}$. The graph G_x is λ-regular with $v-1$ vertices. We first classify all nonisomorphic λ-regular graphs of order $v-1$. Although multiple edges are allowed, however if we are only concerned with simple designs, then merely simple graphs would be of interest. We retain the representation vectors of graphs (which are of dimension $\binom{v-1}{2}$) in the canonical form. Let denote these vectors by E_i $(1 \leq i \leq s, s$ is the number of graphs) such that $E_1 > E_2 > \cdots > E_s$. We then try to find a complete invariant to distinguish between these graphs. For small values of v and λ, there are usually a handful of λ-regular graphs of order $v-1$ and it would not be hard to find such an invariant. We will see that for our purpose it suffices to count the number of cycles of length 3,4,5, etc. Finally, we have to compute and retain the automorphism group of each graph. Note that for the sake of simplicity in the implementation of the algorithm, we consider these automorphisms as permutations on $X \setminus \{1\}$ instead of $X \setminus \{v\}$.

The backtracking algorithm constructs the solution vector D stage by stage. At any stage of the algorithm the following is carried out: An entry of D is set to a suitable value, following it the presetting procedure is called to preset some entries of D by looking at the rows of W^v and at last the canonicity of the partial vector D is tested.

We now explain our method for the canonicity test. We remind that a vector solution D is in the canonical form if $D \geq \sigma D$ for all $\sigma \in S_X$. Therefore, in order to check the canonicity of D, we must examine all permutations on X. However, we do not need to wait until D is completed (i.e. all of its entries are determined and so D represents a design) by the backtracking algorithm. Any permutation σ with $\sigma(x) = 1$ can be taken into account for the canonicity test during the backtracking process when the neighborhood graph G_x associated to point x is completed. This is since with the completion of G_x, σ will provide valuable information about the canonicity of D. Having this in mind, we present the following procedure for the canonicity test.

Suppose that we are at the end of stage $i \geq 1$ of the backtracking where the presetting process is completed on the partial vector D and it is time to test the canonicity of D. From the previous stage of the backtracking, we have a set P_{i-1} of stored permutations on X (note that we set $P_0 = \varnothing$). Now let $P_i := P_{i-1}$. There may be a need to augment some other permutations to P_i, namely those for which $x \to 1$, if the neighborhood graph G_x associated to some point x is completed at the current stage i. So suppose that G_x is completed at the current stage. Let the representation vector of the neighborhood graph G_1 (associated to the point 1) be E_f (note that G_1 is in the canonical form). Using the previously mentioned invariants, we determine r such that the representation vector of G_x is isomorphic to E_r. If $r < f$, then it is clear that the partial vector D constructed so far does not lead to a canonical form and so we backtrack to the stage $i - 1$. If $r > f$, then for any permutation σ with $\sigma(x) = 1$, we have $D < \sigma D$ and therefore σ must be omitted from the canonicity test hereafter. Now suppose that $r = f$. In this case we have to update P_i using permutations with $x \to 1$. G_x is isomorphic to G_1 and we have to determine all the isomorphisms from G_x to G_1. We compute an isomorphism σ from G_x to G_1 (note that σ is a map from $X \setminus \{x\}$ into $X \setminus \{1\}$). Then all the isomorphisms from G_x to G_1 are obtained by composing the automorphisms of G_1 (calculated and retained before the execution of the backtracking algorithm) with σ. These isomorphisms which are from $X \setminus \{x\}$ into $X \setminus \{1\}$ are extended to X by letting $x \to 1$ and then they are augmented to the set P_i. We perform this for all points x for which G_x is completed at the current stage i. Now still being at the stage i of the backtracking, we make use of the elements of P_i to test the canonicity of D as follows. Take an element $\tau \in P_i$, compute τD which is a (possibly) partial vector (since D is partial) and compare τD to D. If $\tau D > D$ (compared partially), then obviously D is not in the canonical form and hence we backtrack to the stage $i - 1$. If $\tau D < D$, then delete τ from P_i. Carry out this procedure for any element of P_i until a backtrack occurs or we examine all the elements of P_i. If the latter happens, then we check if D is a complete vector (all of its entries are determined) and if so, then we have found a design with P_i constituting all of the automorphisms of D and after saving the necessary data (for example the design itself, the number of its automorphisms or anything about the desired design) we backtrack to the stage $i-1$, otherwise we proceed to the stage $i+1$ of the backtracking algorithm retaining P_i for the later use.

We note in passing that the canonical form defined here and also the notion of neighborhood graphs have previously been used in the enumeration of triple systems [1, 12, 20]. In these references the graphs are employed to perform partial isomorph rejections and hence it is necessary to use a design isomorphism program to filter isomorphic copies at the final stage. In our algorithm we are sure that the completed designs are all mutually nonisomorphic and hence there is no need for a separate program to eliminate isomorphic copies.

3 2-(8,3,6) designs

By [22], the exact number of 2-(8,3,6) designs is 3,077,244 which is attributed to E. Spence. To our knowledge, Spence has not published this result. We use our method to confirm this classification. There are 955 6-regular multigraphs on 7 vertices. In order to identify these graphs, we count the number a_i of cycles of length i for $2 \leq i \leq 7$. These numbers are enough to distinguish all graphs except for two. Those are graphs numbered 924 and 928 in the lexicographical ordering of the canonical forms which runs from 1 to 955. For these exceptional cases we made use of the isomorphism test routine which was a part of the main program to establish isomorphism among graphs. We summarize the results in Table 2.

Table 2 2-$(8,3,6)$ designs

#Automorphisms	#Designs	#Automorphisms	#Designs
1	3,056,668	16	10
2	19,078	20	4
3	555	21	3
4	664	24	13
6	129	32	2
7	6	48	4
8	77	64	1
10	1	120	1
12	23	168	1
14	3	40320	1

The number of distinct blocks in a 2-(v,k,λ) design is called the *support size*. In [6, 13], for any $22 \leq s \leq 50$ and $s = 52, 56$, a 2-(8,3,6) design with support size s is given and in [14], it is proven that 22 is the minimum possible support size for these designs. For any generated design by our program, we have retained the support size. Table 3 presents the number of designs for any possible support size. As it is seen from the table, there is a unique design with each of the support sizes 22 and 52.

Table 3 Support sizes for 2-$(8,3,6)$ designs

Support size	#Designs	Support size	#Designs
22	1	38	257,921
23	17	39	166,916
24	86	40	95,176
25	405	41	47,430
26	1,827	42	21,014
27	5,974	43	8,120
28	16,722	44	2,821
29	39,479	45	829
30	82,062	46	259
31	148,449	47	55
32	235,326	48	20
33	330,494	49	3
34	406,966	50	4
35	439,443	52	1
36	419,400	56	1
37	350,023		

4 2-(11,3,3) designs

We have employed the algorithm to classify all simple 2-$(11,3,3)$ designs. There are 135 3-regular multigraphs on 10 vertices from which 21 graphs are simple. In order to distinguish these graphs, we count the number a_i of cycles of length i and it turns out that these numbers for $3 \leq i \leq 5$ ($2 \leq i \leq 6$) uniquely determine each of the simple graphs (multigraphs). We save the graphs in the canonical form along with their automorphism groups and the number of cycles. These data are used by the main program. The algorithm was implemented and run on the cluster of 14 Pentium 4 CPU 2.6 GHz. After the completion of running in about 2 days we found a total of 7,038,699,746 nonisomorphic simple 2-$(11,3,3)$ designs. In Table 4, we give the number of designs for any possible automorphism group order.

Table 4 Simple 2-$(11, 3, 3)$ designs

#Automorphisms	#Designs
1	7,038,452,602
2	239,601
3	6,811
4	398
5	85
6	198
8	3
10	12
11	4
12	26
24	2
60	1
72	1
110	1
660	1
Total	7,038,699,746

5 2-(12,3,2) designs

There exist exactly 242,995,846 2-(12,3,2) designs. This result was obtained in [25]. We use our program to confirm this classification. There are 14 2-regular multigraphs on 11 vertices out of which 6 graphs are simple. In order to identify these graphs, we count the number a_i of cycles of length i and it turns out that these numbers for $2 \leq i \leq 5$ uniquely determine each of the multigraphs. We save the graphs in the canonical form along with their automorphism groups and the numbers of cycles. These data are used by the main program. The execution time of program was about 2 hours on the cluster. Consequently, the correctness of Table 1 of [25] was established.

6 Steiner triple systems of order 19

The problem of classification of Steiner triple systems of order 19 (i.e. 2-(19,3,1) designs) was open for quite a long time until it was settled by Kaski and Östergård [17]. Prior to [17], many partial results appeared in the literature. Among these, it is worth mentioning [2] and [28] which dealt with the classification of systems with nontrivial automorphism groups and systems containing a subsystem of order 9, respectively.

The method used in [17] is an orderly algorithm. First all possible partial systems containing blocks through one of the three initial fixed points are classified. Then these partial systems are completed using the exact cover algorithm. In the final step, the package *nauty* and an invariant based on Pasch configurations are used to reject isomorphic copies.

We use a modified version of the algorithm described in Section 2 to confirm this classification. In Steiner triple systems, the neighborhood graph of any point is just a matching and it is not very useful in the algorithm. Instead we consider the neighborhood graphs associated to pairs of distinct points. Let (X, B) be a 2-$(v, 3, 1)$ design. For any pair of distinct points $x, y \in X$, we define its neighborhood graph to be the graph with the vertex set $V = X \setminus \{x, y, u\}$ and the edge set $E = \{\{z, t\}|\ z, t \in V, \{x, z, t\}$ or $\{y, z, t\} \in B\}$ where $\{x, y, u\} \in B$. This is a simple 2-regular graph of order $v-3$ which is decomposable into two disjoint one-factors (and therefore it is a union of disjoint even cycles). The modified algorithm uses these graphs (instead of neighborhood graphs of points) in the canonicity tests. The rest of algorithm is the same as in Section 2. Some minor changes in the computer program for triple systems is needed in order to make it applicable in classifying Steiner triple systems.

There are seven 2-regular graphs of order 16 which are decomposable into two disjoint one-factors. These correspond to the following partitions of 16:

$$16, \quad 12+4, \quad 10+6, \quad 8+8, \quad 8+4+4, \quad 6+6+4, \quad 4+4+4+4.$$

For example, 12+4 corresponds to the union of disjoint cycles of lengths 12 and 4. The numbers of cycles of lengths $i \leq 16$ can be used to distinguish these graphs from each other. We retain the graphs in the canonical form along with their automorphism groups and the numbers of cycles. These data are used by the main program. The program ran for approximately three months on the cluster of 14 Pentium 4 CPU 2.6 GHz. Our results confirm those reported in [17].

Acknowledgments The authors thank the referees for their careful reading and valuable comments which improved the presentation of the paper.

References

[1] C. J. Colbourn, M. J. Colbourn, J. J. Harms and A. Rosa, A complete census of $(10, 3, 2)$ block designs and of Mendelsohn triple systems of order ten, III. $(10, 3, 2)$ block designs without repeated blocks, *Congr. Numer.* **37** (1983), 211–234.

[2] C. J. Colbourn, S. S. Magliveras and D. R. Stinson, Steiner triple systems of order 19 with nontrivial automorphism group, *Math. Comp.* 59 (1992), 283–295.

[3] F. N. Cole, The triad systems of thirteen letters, *Trans. Amer. Math. Soc.* **14** (1913), 1–5.

[4] F. N. Cole, L. D. Cummings and H. S. White, The complete enumeration of triad systems in 15 elements, *Proc. Nat. Acad. Sci. U.S.A.* **3** (1917), 197–199.

[5] I. A. Faradžev, Constructive enumeration of combinatorial objects, *Problmes combinatoires et thorie des graphes* (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976), pp. 131–135, Colloq. Internat. CNRS, 260, CNRS, Paris, 1978.

[6] W. Foody and A. S. Hedayat, On theory and applications of BIB designs with repeated blocks, *Annals of Statistics* **5** (1977), 932–945.

[7] B. Ganter, A. Gülzow, R. Mathon and A. Rosa, A complete census of $(10,3,2)$ block designs and of Mendelsohn triple systems of order ten, IV. $(10,3,2)$ block designs with repeated blocks, Math. Schriften Kassel 5/78 (1978).

[8] P. B. Gibbons and P. R. J. Östergård, Computational methods in design theory, in: *Handbook of Combinatorial Designs* (C. J. Colbourn and J. H. Dinitz, eds.), Second Edition, Chapman & Hall/CRC Press, Boca Raton, FL, 2007, pp. 755–783.

[9] S. W. Golomb and L. D. Baumert, Backtrack programming, *J. Assoc. Comput. Mach.* **12** (1965), 516–524.

[10] T. Grüner, R. Laue and M. Meringer, Algorithms for group actions applied to graph generation, *Groups and computation, II* (New Brunswick, NJ, 1995), 113–122, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., 28, Amer. Math. Soc., Providence, RI, 1997.

[11] M. Hall, Jr. and J. D. Swift, Determination of Steiner triple systems of order 15, *Math. Tables Aids Comput.* **9** (1955), 146–152.

[12] J. J. Harms, C. J. Colbourn and A. V. Ivanov, A census of $(9,3,3)$ block designs without repeated blocks, Sixteenth Manitoba conference on numerical mathematics and computing (Winnipeg, Man., 1986), *Congr. Numer.* **57** (1987), 147–170.

[13] A. S. Hedayat and H. L. Hwang, BIB(8, 56, 21, 3, 6) and BIB(10, 30, 9, 3, 2) designs with repeated blocks, *J. Combin. Theory Ser. A* **36** (1984), 73–91.

[14] A. S. Hedayat, I. N. Landgev and V. D. Tonchev, Results on the support of BIB designs, *J. Statist. Plann. Inference.* **22** (1989), 295–306.

[15] A. V. Ivanov, Constructive enumeration of incidence systems, *Ann. Discrete Math.* **26** (1985), 227–246.

[16] P. Kaski and P. R. J. Östergård, *Classification Algorithms for Codes and Designs,* Number 15 in Algorithms and Computation in Mathematics, Springer-Verlag, Berlin Heidelberg, 2006.

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

[18] G. B. Khosrovshahi, M. Mohammad-Noori and B. Tayfeh-Rezaie, Classification of 6-$(14, 7, 4)$ designs with nontrivial automorphism groups, *J. Combin. Des.* **10** (2002), 180-194.

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

[20] R. Mathon and D. Lomas, A census of 2-$(9, 3, 3)$ designs, *Australas. J. Combin.* **5**(1992), 145–158.

[21] R. Mathon, K. T. Phelps and A. Rosa, Small Steiner triple systems and their properties, *Ars Combin.* **15** (1983), 3–110.

[22] R. Mathon and A. Rosa, On 2-(v, k, λ) designs of small order, in: *Handbook of Combinatorial Designs* (C. J. Colbourn and J. H. Dinitz, eds.), Second Edition, Chapman & Hall/CRC Press, Boca Raton, FL, 2007, pp. 25–58.

[23] B. D. McKay, Isomorph-free exhaustive generation, *J. Algorithms* **26** (1998), 306–324.

[24] M. Mohammad-Noori and B. Tayfeh-Rezaie, Backtracking algorithm for finding t-designs, *J. Combin. Des.* **11** (2003), 240–248.

[25] P. R. J. Östergård, Enumeration of 2-$(12, 3, 2)$ designs, *Australas. J. Combin.* **22** (2000), 227–231.

[26] V. de Pasquale, Sui sistemi ternari di 13 elementi, *Rend. Reale Ist. Lombardo Sci. Lett. Ser.* 2 **30** (1899), 213-221

[27] R. C. Read, Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations, *Ann. Discrete Math.* **2** (1978), 107–120.

[28] D. R. Stinson and E. Seah, 284 457 Steiner triple systems of order 19 contain a subsystem of order 9, *Math. Comp.* **46** (1986), 717–729.

[29] R. J. Walker, An enumerative technique for a class of combinatorial problems, 1960 Proc. Sympos. Appl. Math., Vol. 10, pp. 91–94, American Mathematical Society, Providence, R.I.

[30] H. S. White, F. N. Cole and L. D. Cummings, Complete classification of triad systems on fifteen elements, *Memoirs Nat. Acad. Sci. U.S.A.* **14** (1919), 1–89.