1989

Computing the chromatic number of t-(v,k,[lambda]) designs

Nancy M. Schornstein

Rochester Institute of Technology
School of Computer Science

Computing the Chromatic Number of t-(v,k,λ) Designs

by

Nancy M. Schornstein

A thesis, submitted to
The Faculty of the School of Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science.

Approved by:

19 May 1989

Dr. Donald L. Kreher (Supervisor)

Dr. Stanislaw P. Radziszowski

Dr. Peter G. Anderson

May 19, 1989

Computing the Chromatic Number of t-(v,k,λ) Designs

a thesis, submitted by

Nancy M. Schornstein

Nancy M. Schornstein

Date: May 19 1989

Computing the Chromatic Number of t-(v,k,λ) Designs

Nancy M. Schornstein

ABSTRACT

Colouring t-designs has previously been shown to be an NP-complete problem; heuristics and a practical algorithm for this problem were developed for this thesis; the algorithm was then employed to find the chromatic numbers of the sixteen non-isomorphic $2-(25,4,1)$ designs and the four cyclic $2-(19,3,1)$ designs. This thesis additionally examines the existing literature on colouring and finding chromatic numbers of t-designs.

TABLE OF CONTENTS

1. Introduction 2

2. Background from Relevant Research 6

3. Explanation of the Colouring Algorithm 12

4. Preparation of Input Data 19

5. Explanation of Solutions 22

6. Solutions for the S(3,4,25) designs 23

7. Solutions for the S(2,3,19) designs 28

8. Concluding Remarks 29

Bibliography 33

1. Introduction

A $t-(v,k,\lambda)$ *design* is a pair (X,B) where X is a v-element set of *points* and B is a family of k-subsets of X called *blocks,* such that every t-subset $T \subseteq X$ is contained in exactly λ of the blocks in B. For example: a 2-(7,3,1) design is given by (X,B) where $X=\{1,2,3,4,5,6,7\}$ and $B=\{\{1,2,4\},\{2,3,5\},\{3,4,6\},\{4,5,7\},\{5,6,2\},\{6,7,1\},\{7,1,3\}\}$. This design also happens to be a Steiner system and the notation $S(2,3,7)$ is often used for it. Steiner system is the name given to any design for which $\lambda=1$, which indicates that any unordered t-subset of elements of X is found in precisely one block. Let X be a v-element set. Then a *Steiner triple system of order v, STS(v),* is a collection of triples (3-subsets) of X, such that each unordered pair of elements is contained in precisely one block, and a *Steiner quadruple system of order v, SQS(v),* is a collection of quadruples (4-subsets) of X such that each triple is found in precisely one block. The design shown above is by definition a Steiner triple system. The same design is also a $1-(7,3,3)$ design since any single element is found in precisely $\lambda=3$ blocks where each block is $k=3$ elements in size. It is well known that a $STS(v)$ exists if and only if $v \equiv 1 \text{ or } 3 \pmod 6$ [30], and we call these values of v *admissible orders* of v.

Lindner and Rosa [38] gave a survey of Steiner quadruple system research that includes citings from 1844 -1987. Included in that paper is the reference that Kirkman constructed systems $S(3,4,2^n)$ for every n. Also included in [38] is reference to the fact that Hanani proved the necessary condition $v \equiv 2 \text{ or } 4 \pmod 6$ for the existence of an $S(3,4,v)$ is also sufficient.

Considerable research has been done in the area of t-designs and in Steiner sys-

tems in particular. Applications of Steiner system research in such areas as error-correcting codes, design of experiments and scheduling are well-known[10,40,49]. Research is also ongoing to determine properties of some of the t-designs that are already known to exist. One such property of t-designs, the property of colouring, is the subject of this thesis.

A c-colouring of a block design (X,B) is a mapping $\psi: X \rightarrow \{1,2,3,.....c\}$ such that if the values $\{1,2,3,4...c\}$ are seen as possible colour representations, then a successful colouring would have no monochromatic (single colour) block. More precisely, for every block $K \in B$ there are some $v, v_1 \in K$ with $\psi(v) \neq \psi(v_1)$. We say that the chromatic number of a design is the value c if the design is $c-$*colourable* but not $(c-1)-$*colourable.*

The chromatic number of the 2-(7,3,1) design above is 3. No two-colouring exists since there does not exist an assignment of two colours to the seven vertices of the design in such a way that no single colour block results. A successful 3-colouring is given by $\psi = \begin{pmatrix} 1234567 \\ 1222132 \end{pmatrix}$; Table I shows the colouring induced by ψ.

Table I.

Block	Colouring
1 2 4	1 2 2
2 3 5	2 2 1
3 4 6	2 2 3
4 5 7	2 1 2
5 6 1	1 3 1
6 7 1	3 2 2
7 1 3	2 1 2

Research on colouring and finding chromatic numbers of t-designs is sparse, and the investigation has been restricted to Steiner systems for the most part.

Research on the colouring of t-designs that are not Steiner systems appears to be minimal. In Section 3 an algorithm is presented for determining the chromatic numbers of arbitrary t-designs.

There is a significant body of research on colouring and finding the chromatic numbers of graphs and hypergraphs, [20-25,27,29,31,37,39,41,54-55]. The chromatic number of a graph is the minimum number of colours required to colour vertices of the graph G in such a way that adjacent vertices are coloured differently, i.e. such that there is no monochromatic edge. The edge-chromatic number of a graph is defined analogously. In vertex colouring of graphs, to determine whether a graph is 3-colourable, or to find a 3-colouring, knowing that the graph is 3-colourable is NP-hard, Schmidt [54]. In edge-colouring of graphs finding the chromatic number is solvable in polynomial time for 2-colouring, [57], but Levin and Galil [37] show that it is NP-complete to determine if it is possible to edge-colour a t-regular graph of degree k with k colours for any $k \geq 3$.

Note that t-designs are regular k-uniform hypergraphs and thus are in a sense a generalization of regular graphs. More formally, if $k \geq 1$ is an integer, and X is any set with $|X| \geq k$, $\binom{X}{k}$ denotes the set of all k-subsets of X. A *$k-$uniform hypergraph* G is formed by a set $X(G)$ of vertices and a multiset $E(G)$ of edges such that each edge is an element of $\binom{X(G)}{k}$. This definition of a k-uniform hypergraph allows repeated edges; if a k-uniform hypergraph has no repeated edges, it is called *simple.* The definition of hypergraph is less restrictive than a k-uniform hypergraph in that all of the subsets do not have to be identical in size. A *hypergraph* $H = (X,E)$, has a set X of vertices and a set E of subsets of X. A hypergraph

is a graph if all $e \in E$ have cardinality 2; it is t-regular if each t-subset of vertices is contained in precisely the same number of blocks of the hypergraph.

The terminology of *weak* and *strong* colourings was formalized by Berge in 1973 and has since been used in the context of graph colouring, hypergraph colouring and design colouring, [2]. In *weak* colourings, only monochromatic edges are forbidden, while in *strong* colourings each pair of vertices from the same edge (block) must get distinct colours, [54]. Research is ongoing in both *weak* and *strong* colouring.

The colouring property and other properties of designs are explored primarily for their relevance to existing designs and as an aid to uncovering new designs. Yet, both strong and weak colouring (and many variations between the two) do occur frequently in pactical problems. A universal example for the problem of both strong and weak colouring is the problem of resource allocation that is given in [54].

> The vertices represent the set of users of a system with a number of resources. At any time, a user may request certain resources; the edges represent the set of users, requesting the same resource at the same time. If each resource is physically available to only one user at a time, the problem is a strong colouring problem. (Any two users requesting the same resource at the same time must get distinct colours.) [Thus the strong chromatic number is the number of resources required to meet the different possible sets of user requests.] However if a certain resource has several copies serving the same purpose, (e.g. six identical printers one next to each other), the constraints imposed on the corresponding edge

might be weaker, (e.g. no more than six vertices of that edge are allowed to get the same colour). Depending on the resources the above problem is a 'weak', 'medium' or 'strong' hypergraph colouring problem. The above colouring process has to be repeated over and over, it is therefore natural that one should seek fast algorithms with a good average performance.

By definition this research is in the area of *weak* colouring, although the following section does include references to research that has been recently published in both topic areas. Assume from this point forward that colouring refers to weak colouring unless the word strong is written with it. *Strong* colouring could easily be implemented in the algorithm presented in Section 3 by replacing the statement checking for monochromicity with those that check for distinct colours among vertices in the block.

There are many designs for which the chromatic number may be calculated using the algorithm in Section 3 with either weak or strong colouring. For this thesis the weak chromatic numbers of the sixteen non-isomorphic $2-(25,4,1)$ designs and the four cyclic $2-(19,3,1)$ designs were obtained.

2. Background from relevant research

Although research on colouring t-designs is sparse, there has been a significant amount of research done to find t-designs and Steiner systems. Construction of such designs is a fundamental problem in combinatorial design theory, [28,30,32,36]. The author has elected to include here only references that specifically relate to colouring, since inclusion of design construction research citings would make this section excessive in length.

A partial Steiner triple system relaxes the requirement that any pair of elements be found in exactly one triple by requiring only that any pair of elements be found in at most one triple. Hence in a partial STS(v), you are allowed to consider subsets of the given system that inherit the Steiner property without having to consider the complete set of all of the blocks. Partial t-designs have analogous definitions. It has been shown [24] that for any k, there exists a partial STS(v) that is k-chromatic. It has also been shown that every partial triple system can be embedded in a complete STS(v), and that STS(v) can have arbitrarily large chromatic number, [50,17].

Of particular relevence to colouring t-designs is the research by Colbourn et. al. [8] in which it is proved that deciding whether a t-design is c-colourable is an NP-complete problem when $c \geq 9$. To establish this result Colbourn et. al. first prove in [8] that deciding whether a *partial* STS(v) is n-colourable is NP-complete. Then they show that given a *partial* STS(v) a 2-(12tv+3,3,18tv+3) design can be produced in polynomial time such that the design is 3t-colourable if and only if the *partial* STS(v) is t-colourable. Colbourn et. al. note the importance of this result is in providing the first example of an algorithmic problem in computational design theory that is known to be NP-complete. More recently, it has been shown that deciding n-colourability of STS(v)'s for $n \leq 14$ is NP-complete and also that deciding whether a partial 2-(v,k,1)-design is n-colourable is NP-complete for any $n \geq 2$, [50].

Colbourn, et. al. [9] prove that deciding whether a 3-uniform hypergraph is 2-colourable is NP-complete; in the same paper they also prove that deciding whether a partial SQS(v) is 2-colourable is NP-complete. Additionally, they introduce a greedy selection process for recognizing 2-chromatic Steiner quadruple systems that

relies heavily on implicants, which in turn requires the "2-colour" restriction. That is, the algorithm depends heavily on given vertices being forced to receive an opposite colour. A vertex z is an implicant in a colouring of an SQS(V,E) whenever three vertices w,x,y are coloured the same colour, since it is guaranteed there exists an edge {w,x,y,z} and moreover that z must be given the opposite colour, [9]. The effect of the heavy reliance on implicants limit the effectiveness of the algorithm to only 2-colour trials. On $SQS(2n)'s$, their algorithm completes its work in polynomial time.

The algorithm which has been developed for this thesis and is presented in Section 3 is not limited to 2-colouring, or limited to Steiner systems for its implementation.

N. Alon and Z. Bregman [1] recently presented a proof that every d-uniform d-regular hypergraph is 2-colourable, provided $d \geq 8$. If we let

$$\delta = min\ \{d : \text{Every } d-\text{uniform } d-\text{regular hypergraph is 2-colourable}\},$$

then the above result and the example given in Table I shows that $4 \leq \delta \leq 8$.

Alex Rosa in 1970 [52] established that there exists a 3-chromatic STS(v) of all admissable orders v, $v \geq 7$. Alex Rosa in the same paper gave some constructions for 4-chromatic STS(v)'s, including those for v=31 and v=49. M. Brandes et. al. [17] showed that there exists a 4-chromatic STS(v) for all $v \geq 25$ where $v \equiv 1$ *or* 3 $(mod\ 6)$, except possibly for v=39, 43 and 45, and that there exists a 4-chromatic STS(v) for all $v \geq 49$.

A. Rosa provides an excellent survey article on colouring problems in combinatorial designs, [50]. Included in this survey article are the following theorems and

remarks on colouring and their supporting references.

On weak colouring:

Theorem 1 [50]: For every $n>3$, there is a positive integer $v(n)$ such that for all admissible orders $v \geq v(n)$ there exists an n-chromatic STS(v) and

$$c_1 n^2 \ln n \leq v(n) \leq c_2 n^2 \ln n.$$

Alex Rosa proved that $v(3)=7$ in [52] (See Table I.) and it was conjectured that $v(4)=25$, but only proven that $v(4) \leq 49$, [17].

Theorem 2 [50]: If S is an STS(v) then the weak chromatic number $\chi_W(S)$, is given by $\chi_W(S) \leq c\sqrt{(v / \ln v)}$ where c is an absolute constant.

Theorem 3 [50]: Deciding whether a partial SQS(v) (Steiner Quadruple System) is 2-colourable is NP-complete while deciding whether a SQS(v) is 2-colourable can be done in polynomial time.

On strong colouring:

The strong chromatic number $\chi_S(S)$ of any $t-(v, k, \lambda)$ design with $t \geq 2$ is equal to v since every pair of elements appear in a block, v colours are needed, and suffice. However this is not the case for partial designs, and strong vertex colourings of partial STS(v)'s are the subject of recent investigation by Colbourn and others, [11].

Theorem 4 [50]: Deciding whether a partial STS(v) has a strong n-colouring is NP-complete for any $n \geq 3$.

Several other results on colouring partial Steiner triple systems are found in Alex Rosa's paper [50], and in other papers cited, but are not specifically listed here

since the focus is on colouring full, as opposed to, partial designs. Included in Rosa's paper are some problems which he poses for future solution.

Investigations of the chromatic number (of hypergraphs) have led both directly and indirectly to elegant constructions for t-designs; one recent example is the investigation of 2-chromatic SQS(v), [9,48].

For reference to colouring random hypergraphs, see Schmidt [54] which presents colouring algorithms for several strong colouring problems, and analyzes their performance in spaces of random hypergraphs, and [55] which makes a corresponding effort for weak colouring.

To this point in the paper, colouring the vertices of t-designs and Steiner systems has been the primary focus of the discussion.

Another form of colouring is block colouring. Given a $t-(v,k,\lambda)$ design $D=(X,B)$, a *block−colouring* of D is a mapping $\psi:B\rightarrow C$ such that for $b,b_1\in B$, $\phi(b)=\phi(b_1)\rightarrow b\cap b_1=\oslash$. If $|C|=n$, we have an *n−block−colouring.* The *chromatic index* of D, $\chi(D)$ is the smallest n such that there exists an n-block-colouring of D. Since the term *colour class* is given to a set of pairwise disjoint blocks, an n-block-colouring can also be defined as a partition of the block set into n colour classes, [5]. The definition applies analogously to either t-designs or partial t-designs, and is included here to warn the reader that there is a difference between the terms chromatic number which refers to an n-colouring and chromatic index which refers to an n-block-colouring.

Consider the 7 point projective plane, the 2-(7,3,1) design given as the first example in this thesis. Table I shows that the chromatic number of that design is

three. Note that no two blocks of the design are disjoint; therefore, a partition of the design into non-intersecting colour classes can only be the original design with a different colour assigned to each block. Since there are seven blocks, the chromatic index of the design is therefore seven. Thus the chromatic number and chromatic index may differ for a design.

In 1982, Charles Colbourn used a branch and bound algorithm in computing the chromatic index of Steiner Triple Systems of order 15, finding that thirteen had chromatic index eight, and sixty-three had chromatic index ninei, [5]; Cole had already determined that the remaining four had chromatic index seven, [5]. By dividing the STS(15)'s into n colour classes, Colburn was able to isolate sub-systems. That is, if a STS(15) were to represent all possible matchings of pairings in tournament play with 15 players, the colour classes or sub-systems of the design that Colbourn identified could be run in parallel with the results of one colour class not affecting the others, and the total time of the tournament being the smallest value of n for which such a partitioning into colour classes exists. The benefit of this partitioning is in providing a mechanism for implementation of parallelism, that did not exist before the partitioning.

Significant in Colbourn's work with computing the chromatic index of STS(15)'s is the use of a branch and bound algorithm to eliminate from consideration excess cases. Once a test case was found to fail, all extensions of that case were eliminated and not considered further. He also employed a priority mechanism for choosing which case to extend; since optimal colouring would use all blocks and as few colours as possible, the priority mechanism he used was the maximum $p-vc$ where p is the number of blocks in the partial colouring, c the number of

colours, and v the number of elements in the design. Two simple heuristics were also used; the first involved ignoring all partial colourings using as many or more colours than the successful colouring once that colouring completed, and the second chose to extend a colouring by selecting the next block to extend to be the block that could be assigned the fewest number of colours, [5].

Chromatic index and chromatic number are different problems each with different applications, but a combinatorial algorithm for determining either of them can benefit from a fast efficient algorithm for generating the other.

Ideas have been taken from Charles Colbourn's work on chromatic index, and applied with variations to find the chromatic numbers of designs.

3. Explanation of the Colouring Algorithm

The work by Charles Colburn [5] on the chromatic index of Steiner triple systems gave the impetus to the thesis you are reading. After investigating designs and Steiner systems, algorithm COLOUR for finding the chromatic number of a specific class of blocksize 4 designs, the sixteen non-isomorphic Steiner systems, $2-(25,4,1)$, [33,34], was developed and then adapted to work on other designs with different design parameters. The chromatic number for each of the $2-(25,4,1)$ was not known when this thesis was begun. This result and others are found in Section 6 and 7 of this thesis.

Before explaining algorithm COLOUR which was developed for colouring, heuristics and speed-up strategies used will be explained. To begin, the algorithm had to be generalizable to designs with different blocksizes, number of vertices in the design and number of times a vertex was found in the blocks of the design.

This led to a decision not to use the property inherent in $S(t,k,\lambda)$ Steiner systems that every t-element subset of vertices occur only once in the design. Instead, the algorithm was designed in such a way that it could colour any t-design, not just Steiner systems.

Therefore, as initialization to the colouring, the input design is analyzed to find both the occurrence count (the number of times each vertex is found anywhere in the design) and the dual design (which for each vertex lists the blocks that contain that vertex). Both the occurrence count and the dual design are used for reference during program execution so that the search to locate a vertex within blocks becomes only a table look up.

The ideas just described were discovered while the author was considering how to quickly colour *partial* Steiner systems, wherein the occurrence count for vertices could vary widely since sets of vertices therein are required to occur no more than one time and are not "required" to occur at all. Therefore the number of times a vertex occurs and where it occurs in a partial system may be very different for each $v \in X$, and it would be necessary to maintain this information to colour the partial system.

The first version of algorithm COLOUR was implemented using recursive methods, but the code was rewritten without recursion to eliminate the overhead spent on the stack operations that are an unwanted side-effect of each recursive call. Speed of execution was a prime factor in implementation, knowing that as the number of vertices v in designs grew, the possible colourings to be examined grew exponentially. Not wanting to examine each possibility of colourings, branch and

bound techniques on a priority queue were used to choose most promising partial colourings for extension and to eliminate failed partial colourings.

Priority values were assigned to each possible partial colouring, with highest values given to colourings which were the best candidates for extension to a complete colouring of the input design. Many priority schemes were examined, starting with the one Colbourn used [5] that is detailed in Section 2 of this paper.

Colouring a design with a large number, η, of colours is likely to be successful and to complete colouring early, since the possibility of finding a monochrome block is roughly inversely proportional to the number of colours being used. The author developed a heuristic that significantly decreases the possible colourings to examine by finding one successful η colouring, deleting from consideration all existing partial colourings that use number of colours $N \geq \eta$ colours, and implementing a new upper bound of $\eta - 1$ on all future N. Determining chromatic number necessitates finding successful colourings with minimal N. The heuristic used here of backing into the smaller number of colours from a larger number differs from Colbourn's techniques which builds from the minimal number of colours, adding a new colour to a block-colouring if it is needed to successfully complete a colouring. Al Biles of Rochester Institute of Technology inspired the idea of backing into minimal colouring when he said in an AI class that decreasing the size of a search space by ten percent is more effective than speeding up a search of the space by ten percent.

A priority scheme to implement this method for decreasing the number of cases would give highest values to partial colourings that complete fastest, and this can be done with priority = (number of blocks coloured) * (number of colours

used). When a successful colouring is found, you prune away all partial and full colourings that use the same or a larger number of colours, and need only consider colourings that use fewer colours. The implementation used set an active chromatic number as a bound for the highest number of colours to be assigned in any colouring, and initially assigned to this bound a value which was expected to result in a successful complete colouring. When a successful colouring was found, the bound was decreased by 1 and colourings that used a number of colours less than or equal to the new bound were attempted.

Colouring one vertex means establishing a colour for one part of each of several blocks. This fact led me to the heuristic implemented of establishing immediately the overall effect of colouring one block. If you pick any block and choose to colour it, the set of vertices found in that block receive some combination of colours. For example if a block to colour is $\{1, 4, 0\}$, then possible 2-colourings for this block could be represented as $\{1,1,1\}$, $\{1,1,2\}$, $\{1,2,1\}$, $\{1,2,2\}$, $\{2,2,1\}$, $\{2,2,2\}$. Since monochrome blocks are not allowed in a proper colouring $\{1,1,1\}$ and $\{2,2,2\}$ are dropped and $\{1,1,2\}$, $\{1,2,1\}$, $\{1,2,2\}$, $\{2,2,1\}$ remain. Therefore, you have four possible partial colourings that you could extend by colouring another block. Checking these four partial colourings immediately may show that some or all of them may have just created a monochrome block somewhere else in the design. Checking for monochromicity as each possibility of block colouring is generated significantly reduces your search space and hence your execution time. In effect, you are pruning a failed case as quickly as a failure of any kind is genereated. Evaluation of failure can be implemented very efficiently by examining only blocks that are already completely coloured and bypassing all partially-coloured or

uncoloured blocks.

To implement the branch part of the branch and bound, the queue entry that had the highest priority value (maxpriority) was kept and a previous maximum as well; these were set each time a successful new colouring received its priority value, thus eliminating nearly all searches for the best priority value. The "best" or "most-likely-to-succeed" colouring was thus extended on the next iteration of the loop. Even when the current partial colouring failed, the previous best was automatically extended on the next loop iteration. Each colouring that was found to be unsatisfactory was removed from the queue so that all possible extensions of that colouring were not examined, hence eliminating from consideration a large number of cases and significantly reducing the computing time, ie. as soon as a colouring failed the remaining possible colourings for the rest of its blocks were ignored.

Another speed-up technique used was to consider for colouring only those blocks that were almost coloured, ones that had the fewest number of vertices left uncoloured. This means that blocks that were fully coloured as a result of colouring done on other blocks were never selected for colouring; they did not need to be selected since they were already fully coloured. Further, blocks could often be coloured with only one vertex needing colour, adding a maximum of $c-1$ entries to the queue where c is the number of possible colours for a vertex. This is obviously an improvement over selecting an arbitrary next block which could add (blocksize * $c-1$) entries to the queue, and possibly take much longer before monochrome blocks are found for each of the new entries.

3.1 The Algorithm.

In this section pseudocode for algorithm COLOUR is exhibited.

Definitions and Notations used in the algorithm COLOUR

b = number of blocks in the design.

r = number of blocks containing a given vertex.

v = number of vertices in the design.

k = blocksize used in the design.

len = length of priority queue.

The priority queue Q is maintained as five arrays: vertex, block, c, B, and p.

vertex[i, h] is the colour assigned to vertex h for partial colouring i; 0 if no colour has been assigned.

block[i, j] is the number of vertices in the j-th block of the design that have some colour assigned to them by the i-th partial colouring.

c[i] = max{vertex[i, h] : h = 1,2, ...v} and is the number of distinct colours used in the i-th partial colouring.

B[i] = |{j : block[i, j] = k }| and is the number of blocks of the design that have been fully coloured.

p[i] = priority value assigned to the i-th colouring

q = pointer to current priority queue entry

max = number of Q entry that has maximum priority value

pmax = number of Q entry that has second best priority value

ψ = best guess at chromatic number; quess at least one more than what you expect the chromatic number will be.

LOWVALUE = constant to show partial colouring has failed

dual[i, j] the ith block contains vertex j of the design, i=1,2,...r.

Algorithm COLOUR:

p[0] ← 1

p[i] ← LOWVALUE for i = 1,2,3,...len

vertex[0, h] ← 0 for h = 1,2,3,...,v

block[0, j] ← 0 for j = 1,2,3,...,b

max ← 0

pmax ← 0

Place in the j-th column of dual[i, j] the blocks that contain the j-th vertex of the design.

Loop forever

q ← max

If p[q] = LOWVALUE

q ← pmax

If p[q]=p[pmax]=LOWVALUE

Find, if possible, max and pmax such that p[max] ≥ p[pmax] > LOWVALUE are as large as possible. If there is no such value for max, search of colourings is complete so exit.

q ← max

End If

End If

/* You now have a partial colouring entry q to extend */

Find j such that block[q, j] < k is as large as possible.

If j is found, colour the uncoloured vertices of the j-th block, block[q, j], as follows:

firstimeinblock ← true

Increment value of B[q]

Use dual[i, x] to identify and update each block[q, n] where n represents only those blocks that contain the same vertices which are now to be coloured in the j-th block.

For (each possible colour assignment τ to the collection of uncoloured vertices [colours ≤ ψ] of the j-th block)

If [firstimeinblock = false and p[q] ≠ LOWVALUE]

/* Add new entry to Q */

q_1 ← smallest i such that p[q] = LOWVALUE

vertex[q_1, h] ← vertex[q, h] for h= 1,2...v

block[q_1, j] ← block[q, j] for j= 1,2...b

B[q_1] = B[q]

q = q_1

End If

firstimeinblock ← false

Update vertex[q, x] by τ

If τ causes a monochrome block

p[q]← LOWVALUE

Else

c[q] ← max {j : vertex[q][i] = j}

```
                p[q] ← c[q] * B[q]
                If p[q] ≥ p[max]
                    pmax ← max
                    max ← q
                End If
        Else (No block j with uncoloured vertices found → colouring complete.)
            Print solution
            ψ = ψ − 1
            If ψ = 1, exit program: successful 2-colouring found.
            For all entries n of Q where c[n] ≥ ψ, p[n] ← LOWVALUE
            End If
    End Loop

End COLOUR
```

4. Preparation of Input Data

Data generation for this thesis, although not key to the thesis results, took a significant amount of time and effort –

- in understanding group theory concepts as they relate to t-designs,
- in digesting and changing the code in three programs written by other graduate students for creating orbit representatives, generating orbits, and converting orbits to block form,
- in generating data using automorphism groups and their orbits over vertices of the design
- and finally in writing programs to reformat STS(v) data to generate the blocks of the STS(v) designs.

To understand how the data was generated, first let me give some relevant definitions. A group $G \leq Sym(X)$ is an *automorphism group* of a $t-(v,k,\lambda)$ design

(X,B) if every $g \in G$ preserves B. That is for all $g \in G$ and $K \in B$ the k-set $K^g = \{x^g : x \in K\}$ is also a block in B. Here x^g denotes the image of x under the generator g, and K^g is said to be the image of the k-set B. The collection $\Gamma = K^G = \{K^g : g \in G\}$ is said to be a G-orbit. Clearly, G is an automorphism group of a $t-(v,k,\lambda)$ design (X,B) if and only if B is a union of G-orbits. The *full automorphism group* of a $t-(v,k,\lambda)$ design (X,B) is the set of all automorphism in $Sym(X)$; the full symmetric group, preserving B.

The authomorphism groups of the sixteen non-isomorphic $2-(25,4,1)$ designs and the four cyclic $2-(19,3,1)$ designs had to be generated for this thesis. The generator function $v \equiv v+1 \,(mod\, 19)$ was used to generate the the four cyclic $2-(19,3,1)$ designs.

The generators used to generate the S(2,4,25)'s in Section 6 are listed in Table II with a variable name at the far left to identify each generator, [33,34].

Table II.

α	(1 2 3)(4 5 6)(7 8 9)(10 11 12)(13 14 15)(16 17 18)(19 20 21)(22 23 24)
β	(1 2 3)(4 5 6)(7 8 9)(10 11 12)(13 14 15)(16 17 18)(19 20 21)
$\hat{\alpha}$	(1 2 3)(4 5 6)(7 8 9)(10 11 12)(13 14 15)(16 17 18)(22 23 24)
$\hat{\beta}$	(1 4 7)(2 5 8)(3 6 9)(10 13 16)(11 14 17)(12 15 18)(19 20 21)
γ_1	(1 2 24 12 8 18 19 9 7 23 17 4 14 21 5 6 22 13 3 10 20)(11 16 15)
γ_2	(1 21 13 18 4 15 2 5 19 17 10 8 16 6 9 20 12 14 3 11 7)(22 23 24)
γ_3	(1 23)(2 24)(3 0)(4 21)(5 22)(6 17)(7 18)(8 19)(9 20)(10 16)
γ_4	(1 0 5)(2 19 10)(3 13 15)(4 7 20)(6 21 24)(8 14 9)(11 22 18)(12 16 23)
γ_5	(1 2 3 4 5 6 7)(8 9 10 11 12 13 14)(15 16 17 18 19 20 21)
γ_6	(1 2 4)(3 6 5)(8 9 11)(10 13 12)(15 16 18)(17 20 19)(22 23 24)
γ_7	(1 20 16)(2 10 3)(5 17 18)(4 19 22)(6 9 0)(11 15 14)(7 21 12)(8 13 24)
γ_8	(2 18)(3 5)(4 22)(6 15)(8 24)(9 11)(10 17)(12 21)(14 0)(16 20)

Included below are the orbit representatives of Design 1 of the S(2,4,25)'s in Section 6 only. Justification for its inclusion here is to indicate how one of the

designs was generated; all other designs were similarly generated.

$$\{1\ 2\ 3\ 19\} \quad \{1\ 5\ 9\ 0\}$$

Generators for the automorphism group of Design 1 are $\hat{\alpha}$ and γ_1, that is:

$\gamma_1 = (1\ 2\ 24\ 12\ 8\ 18\ 19\ 9\ 7\ 23\ 17\ 4\ 14\ 21\ 5\ 6\ 22\ 13\ 3\ 10\ 20)\ (11\ 16\ 15)$

$\hat{\alpha} = (1\ 2\ 3)(4\ 5\ 6)(7\ 8\ 9)(10\ 11\ 12)(13\ 14\ 15)(16\ 17\ 18)(22\ 23\ 24)$

Therefore beginning with the orbit representatives $\{1\ 2\ 3\ 19\}$ and $\{1\ 5\ 9\ 0\}$ $\hat{\alpha}$ and also γ_1 would be applied repeatedly until no other blocks are obtained.

The algorithm written here is intended to illustrate how the orbit might be generated. Note that the algorithm as written would be quite slow, and is included here only to demonstrate how orbits are obtained from representatives and generators.

Algorithm ORBIT:

Let *Gen* be the set of generators
Let *Orbit* $\leftarrow B$
$L \leftarrow 1$
$S \leftarrow 0$
Done $\leftarrow$ *False*
While (not *Done*) do
    For each $g \in$ *Gen* and $k \in$ *Orbit* do
        $S \leftarrow S \bigcup k^g$
    End For each
    Orbit $\leftarrow$ *Orbit* $\bigcup S$
    If $|$ *Orbit* $| \leq L$ then *Done* $\leftarrow$ *True*
    $L \leftarrow |$ *Orbit* $|$
End While

5. Explanation of solutions:

The following tabular format is used to represent the results of this thesis. The $S(3,4,25)$ are given using this format in Section 6, while the $S(2,3,19)$ results are given in Section 7.

Design 1. $G = <\hat{\alpha}, \gamma_1>$.

Orbit representatives	{1 2 3 19}, {1 5 9 0}
Chromatic number	3
Colouring	2 1 1 1 2 2 1 1 1 1 3 1 2 1 1 2 3 1 3 1 3 3 3 3

The entry for colouring is a list of numbers that indicates the vertex colouring assignment selected for the twenty-five vertices of the design. Vertex 0 has colour 2, vertices 1 through 4 have colour 1, vertices 5 and 6 have colour 2, etc.

The orbit representatives of the design are listed first in the table while the generators of the design are found above the table as $G = <\hat{\alpha}, \gamma_1>$, indicating that $\hat{\alpha}$ and γ_1 are the generators for the Design 1's automorphism group. Table II in Section 4 [33,34], contains a listing of the contents of generators of the $S(3,4,25)$ designs. To understand how generators and orbit representatives are used, see Section 4, "Preparation of Input Data". Included in Table III is a complete list of the blocks of Design 1 of the $S(3,4,25)$'s with the colouring induced by the colouring entry above. It is easy to see that there is no monochrome block. To save space, the solutions for the remaining designs are printed only as a list of vertex colours; a simple program can verify that each colouring list will yield no monochrome blocks.

Table III.

Block	Colouring
1 2 3 19	1 1 1 3
3 7 11 22	1 1 3 3
8 10 20 22	1 1 1 3
3 15 17 20	1 1 3 1
4 12 16 21	1 1 2 3
4 18 19 24	1 1 3 3
2 14 16 20	1 1 2 1
8 11 13 19	1 3 2 3
5 8 14 15	2 1 1 1
5 10 17 21	2 1 3 3
7 8 9 21	1 1 1 3
6 17 19 23	2 3 3 3
2 13 21 22	1 2 3 3
1 6 14 22	1 2 1 3
0 1 5 9	2 1 2 1
0 3 4 8	2 1 1 1
0 19 20 21	2 3 1 3

Block	Colouring
2 9 10 24	1 1 1 3
10 13 16 23	1 2 2 3
1 13 18 20	1 2 1 1
9 11 20 23	1 3 1 3
1 7 16 17	1 1 2 3
12 15 18 22	1 1 1 3
1 15 21 24	1 1 3 3
9 12 14 19	1 1 1 3
5 16 19 22	2 2 3 3
4 5 6 20	1 2 2 1
6 11 18 21	2 3 1 3
6 9 13 15	2 1 2 1
5 7 18 23	2 1 1 3
4 7 13 14	1 1 2 1
0 2 6 7	2 1 2 1
0 10 14 18	2 1 1 1
0 11 15 16	2 3 1 2

Block	Colouring
7 12 20 24	1 1 1 3
1 8 12 23	1 1 1 3
2 8 17 18	1 1 3 1
11 14 17 24	3 1 3 3
1 4 10 11	1 1 1 3
3 9 16 18	1 1 2 1
7 10 15 19	1 1 1 3
2 4 15 23	1 1 1 3
2 5 11 12	1 2 3 1
6 8 16 24	2 1 2 3
3 6 10 12	1 2 1 1
3 5 13 24	1 2 2 3
3 14 21 23	1 1 3 3
4 9 17 22	1 1 3 3
0 22 23 24	2 3 3 3
0 12 13 17	2 1 2 3

6. Results of S(3,4,25)'s

The designs below are all S(3,4,25), and the data was generated using the generators and representatives found by Kramer et.al., [33,34]. Each of the designs is shown below using the generator form; see Section 5 for explanation of how to interpret the tables below.

Design 1. $G = <\hat{\alpha}, \gamma_1>$.

Orbit representatives	{1 2 3 19}, {1 5 9 0}
Chromatic number	3
Colouring	2 1 1 1 2 2 1 1 1 1 3 1 2 1 1 2 3 1 3 1 3 3 3 3

Design 2. $G = <\hat{\alpha}, \gamma_2>$.

Orbit representatives	{1 2 3 19}, {1 4 7 22}, {19 20 21 0}, {22 23 24 0}
Chromatic number	3
Colouring	2 1 1 1 1 1 2 1 1 2 1 1 2 1 1 2 1 3 3 3 3 3 3 3 3

Design 3. $G = <\hat{\alpha}, \hat{\beta}>$.

Orbit representatives	{1 6 10 11}, {1 12 13 23}, {1 14 16 20}, {1 15 21 24}
	{1 2 3 19}, {1 4 7 22}, {1 5 9 0}, {10 13 16 0}
	{19 20 21 0}, {22 23 24 0}
Chromatic number	3
Colouring	2 1 1 1 1 1 1 1 2 2 1 3 3 1 1 3 2 1 1 3 3 3 3 3 3

Design 4. $G = <\hat{\alpha}, \hat{\beta}>$.

Orbit representatives	{1 6 10 14}, {1 11 13 21}, {1 15 20 24}, {1 16 18 23}
	{1 2 3 19}, {1 4 7 22}, {1 5 9 0}, {10 13 16 0}
	{19 20 21 0}, {22 23 24 0}
Chromatic number	3
Colouring	2 1 1 2 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3

Design 5. $G = <\hat{\alpha}, \hat{\beta}>$.

Orbit representatives	{1 4 10 15}, {1 6 11 22}, {1 13 17 20}, {1 14 21 24}
	{1 2 3 19}, {1 5 9 0}, {10 11 12 0}, {10 13 16 22}
	{19 20 21 0}, {22 23 24 0}
Chromatic number	3
Colouring	2 1 1 2 1 2 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3

Design 6. $G = <\gamma_3, \gamma_4>$.

Orbit representatives	{1 2 6 0}, {1 3 11 19}
Chromatic number	3
Colouring	1 1 1 1 1 1 3 1 1 2 1 3 3 1 2 2 3 2 1 1 3 3 2 1 3

Design 7. $G = <\gamma_5, \gamma_6>$.

Orbit representatives	{1 9 17 22}, {1 2 4 14}, {1 8 15 0}, {1 18 20 21}
	{8 9 11 21}, {22 23 24 0}
Chromatic number	3
Colouring	2 1 1 1 2 1 2 1 2 1 1 1 2 3 1 2 1 1 1 1 3 1 3 3 3

Design 8. $G = <\gamma_7, \gamma_8>$.

Orbit representatives	{1 2 4 0}, {1 3 8 11}, {2 3 14 19}, {2 6 9 12}
	{1 6 7 15}, {1 10 13 17}, {1 12 19 21}, {2 7 18 23}
	{2 17 21 24}, {4 6 8 14}, {4 12 13 24}, {8 9 15 23}
	{1 16 20 23}, {4 19 22 23}
Chromatic number	3
Colouring	1 1 1 1 3 1 1 1 2 1 2 1 2 3 1 2 1 2 1 3 3 3 3 2 3

Design 9. $G = <\alpha>$.

Orbit representatives	{1 4 16 24}, {1 5 12 21}, {1 6 13 22}, {1 7 14 15}
	{1 8 17 18}, {1 9 19 20}, {1 10 11 23}, {4 7 12 19}
	{4 8 9 22}, {4 10 15 18}, {4 13 17 21}, {7 10 13 0}
	{7 11 18 22}, {10 14 16 21}, {13 19 23 24}, {16 19 22 0}
	{1 2 3 0}, {4 5 6 0}
Chromatic number	2
Colouring	2 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 2

Design 10. $G = <\alpha>$.

Orbit representatives	{1 4 20 24}, {1 5 11 23}, {1 6 14 21}, {1 7 15 17}
	{1 8 12 16}, {1 9 13 22}, {1 10 18 19}, {4 7 16 21}
	{4 8 9 23}, {4 11 12 13}, {4 14 17 18}, {7 10 13 0}
	{7 12 19 20}, {10 16 23 24}, {13 15 21 23}, {16 19 22 0}
	{1 2 3 0}, {4 5 6 0}
Chromatic number	2
Colouring	2 1 1 1 1 1 1 2 2 2 1 1 1 2 2 2 2 2 2 1 1 1 2 2 2

Design 11. $G = <\alpha>$.

Orbit representatives	{1 4 19 20}, {1 5 11 23}, {1 6 13 14}, {1 7 15 18}
	{1 8 10 24}, {1 9 16 21}, {1 12 17 22}, {4 7 11 21}
	{4 8 9 23}, {4 12 16 18}, {4 13 17 24}, {7 10 13 0}
	{7 14 16 20}, {10 12 14 21}, {13 21 22 23}, {16 19 22 0}
	{1 2 3 0}, {4 5 6 0}
Chromatic number	3
Colouring	2 1 1 1 1 1 1 1 2 1 1 1 1 2 1 2 3 2 3 1 3 3 3 3 3

Design 12. $G = <\alpha>$.

Orbit representatives	{1 4 20 22}, {1 5 10 18}, {1 6 15 17}, {1 7 12 14}
	{1 8 13 16}, {1 9 23 24}, {1 11 19 21}, {4 7 11 23}
	{4 8 9 21}, {4 10 14 19}, {4 15 16 24}, {7 10 13 0}
	{7 16 17 21}, {10 12 16 23 }, {13 15 19 23}, {16 19 22 0}
	{1 2 3 0}, {4 5 6 0}
Chromatic number	3
Colouring	2 1 1 2 1 1 2 2 1 2 1 1 1 1 1 3 3 1 3 2 1 1 3 3 3

Design 13. $G = <\alpha>$.

Orbit representatives	{1 4 16 23}, {1 5 7 21}, {1 6 12 14}, {1 8 10 18}
	{1 9 13 22}, {1 11 20 24}, {1 15 17 19}, {4 7 22 24}
	{4 8 12 13}, {4 11 19 21}, {4 14 17 18}, {7 8 16 20}
	{7 10 13 0}, {10 11 17 22}, {13 14 19 24}, {16 19 22 0}
	{1 2 3 0}, {4 5 6 0}
Chromatic number	3
Colouring	2 1 1 1 1 1 1 1 1 1 2 2 2 2 1 3 1 3 1 3 3 3 3 3 3

Design 14. $G = <\alpha>$.

Orbit representatives	{1 4 18 23}, {1 5 7 19}, {1 6 12 14}, {1 8 15 24}
	{1 9 11 16}, {1 10 20 22}, {1 13 17 21}, {4 7 22 24}
	{4 8 12 13}, {4 11 19 20}, {4 14 16 17}, {7 8 16 21}
	{7 10 13 0 }, {10 11 17 24}, {13 14 20 24}, {16 19 22 0}
	{1 2 3 0}, {4 5 6 0}
Chromatic number	3
Colouring	2 1 1 1 1 1 3 1 1 1 2 2 1 2 1 1 1 3 1 3 3 3 3 3 3

Design 15. $G = <\beta>$.

Orbit representatives	{1 4 13 24}, {1 5 17 21}, {1 6 8 20}, {1 7 11 0}
	{1 9 16 18}, {1 10 14 15}, {1 12 19 23}, {4 7 10 22}
	{4 8 15 19}, {4 11 12 17}, {4 14 18 0}, {7 12 15 20}
	{7 13 18 23}, {10 17 19 24}, {13 16 19 22}, {1 2 3 22}
	{4 5 6 23}, {7 8 9 24}, {19 20 21 0}, {22 23 24 0}
Chromatic number	3
Colouring	2 1 1 1 1 1 1 2 1 1 2 2 3 1 1 1 1 1 3 3 3 3 3 3 3

Design 16. $G = <\beta>$.

Orbit representatives	{1 4 13 24}, {1 5 16 20}, {1 6 8 19}, {1 7 17 18}
	{1 9 10 0}, {1 11 14 15}, {1 12 21 23}, {4 7 10 22}
	{4 8 14 21}, {4 11 12 16}, {4 15 17 0}, {7 12 14 19}
	{7 15 16 23}, {10 16 21 24}, {13 16 19 22}, {1 2 3 22}
	{4 5 6 23}, {7 8 9 24}, {19 20 21 0}, {22 23 24 0}
Chromatic number	3
Colouring	1 1 1 1 1 1 2 1 1 1 3 3 2 1 1 1 1 2 3 3 3 3 3 3 3

7. Results of S(2,3,19)'s

The designs in this section are the S(2,3,19) designs, each of which was generated cyclically using the same generator function $x \equiv x+1 \ mod\ 19$.

Design 1.

Orbit representatives	{1 4 19}, {2 9 19}, {5 13 19}
Chromatic number	3
Colouring	3 1 1 2 1 2 1 1 1 1 2 3 3 3 2 2 2 3 3

Design 2.

Orbit representatives	{1 4 19}, {2 12 19}, {5 13 19}
Chromatic number	3
Colouring	3 1 1 3 1 2 1 3 1 2 3 2 1 3 2 1 3 2 2

Design 3.

Orbit representatives	{1 12 19}, {2 5 19}, {4 13 19}
Chromatic number	3
Colouring	2 1 1 3 1 1 1 2 3 2 1 3 1 2 3 3 2 3 2

Design 4.

Orbit representatives	{1 12 19}, {2 16 19}, {4 10 19}
Chromatic number	3
Colouring	2 1 1 2 2 1 3 3 3 2 1 2 1 3 2 3 1 3 1

8. Concluding Remarks

Summary of Results

Sections 6 and 7 in this thesis contains tables that indicate the chromatic numbers found using the algorithm developed and implemented by the author. The chromatic number of each of the four cyclic non-isomorphic S(2,3,19)'s is three, while the chromatic number of fourteen of the S(3,4,25)'s is also three; the remaining two (Design 9 & Design 10) of the S(3,4,25)'s have chromatic number of two.

8.1. Problems Encountered and Solved

1. Group generation using orbit representatives and permutations of those representatives were used to generate the data needed for the algorithm. This necessitated understanding groups and their use in design construction, and was accomplished through reading and help provided by my thesis advisor, Dr. Kreher.

2. Computer programs written by other graduate students had to be modified so that they could be used to generate t-designs for input data to the algorithm being developed for colouring. Understanding and changing these programs took substantial time and effort.

3. Two additional programs had to be written by the author that were used to modify input data into a form to be used by the colouring algorithm.

4. Analysis of research done in the area of colouring was completed; this entailed learning to discriminate between colouring problems. The author had to understand graph colouring, design colouring, hypergraph colouring, block-colouring, vertex-colouring, the chromatic index, the chromatic number and the results being

reported within these topic areas in order to be able to categorize those research results and define how those results were applicable to the problem being investigated.

5. In order to develop an algorithm for an NP-complete problem, investigation was done to determine possible implementation strategies which were more efficient than a simple back-track approach. Due to the large number of cases to be examined, efficiency of the implementation was a critical factor. Research led to several possibilities, and the algorithm was written using branch and bound technique. See the Section 4 "Explanation of the Colouring Algorithm" for more details.

6. Partial colourings were assigned priority values to indicate their probable success in being extended to a full colouring. Determination of the formula to use for calculating priority values was a significant and fun part of the project. The author tried several different priority schemes and compared the run time for those schemes, before selecting the fastest running formula to use in the algorithm. The final version of the algorithm has been written in such a way that alternative heuristics for choosing which partial coloring to extend may be easily substituted in the algorithm, and comparisons of computing time made.

7. Significant time was spent trying (both by hand and via computer) to determine counter-examples, e.g. small sets of blocks which could not be 2-colored. If the counter-example, or a set of blocks isomorphic to it, were included in a design then the design would be known to have chromatic number of three or larger; a computer program would need only to locate the counter-example in the design to determine that the design could not be two-colored. This fact becomes a significant

saving as the design grows in size. No counter-example of blocksize 4 was found (yet), though one of blocksize three was already known.

8. Run time of several hours on a time-sharing system was cut to one hour or less in most cases for blocksize 4 designs on 25 vertices with 50 blocks. This was accomplished through program redesign and rewrite.

8.2. Alternative Approaches for Improved System

Variations on the algorithm which could be implemented to possibly improve running time follow:

1. Parallelism could be implemented to complete the analysis of the partial colourings.

2. Bits and bit operations could be used for representation of and manipulation of the colour representations. A decision was made to use integers for colour representations for two reasons:

   a. to easily signify a larger number of colours than two, the usual number of items represented using bit formats, and

   b. to eliminate being restricted by the bit length of an integer, since the number of vertices of the design would be limited to a length less than or equal to the size of an integer as implemented on the machine in use.

It has occured to this author that a scheme for representation of multiple colours for many vertices could be implemented using bitwise operations with just a few integer locations.

8.3. Future Investigation

The following list of problems is currently under investigation by this author.

Find the chromatic number of each of several STS(v) and SQS(v) systems, and then try to determine a classification scheme for determining the chromatic number by mathematical formula.

Determine an algorithm for finding the chromatic index of Steiner systems and other t-designs.

Find the chromatic number of t-designs for which $\lambda \neq 1$, and similarly look for classification and/or formula for determining same.

Investigate correlation between the number of orbit representatives, the number of generators, the orbit length(s) and the chromatic number and chromatic index.

It appears that the integers modulo 6 play an important role in determining existence of Steiner systems. Kirkman showed that $STS(v)$ exist whenever $v \equiv 1,3 \ mod\ 6$, and Hanani showed that $SQS(v)$ exist whenever $v \equiv 2,4 \ mod\ 6$. This author would like to determine the significance of the remaining integers mod 6, i.e. $v \equiv 0,5 \ mod\ 6$.

If the orbit representatives are specified to be mutually exclusive for a design as for example {1 2 3 19} and { 5 6 7 18} would be, calculating the chromatic index could be done by colouring only the orbit representatives; that is the author believes you should be able to determine mathematically the chromatic index without colouring all of the design, simply by carefully selecting the blocks to be coloured.

Determine the relationship between the crossing number of the block-incidence graph and the chromatic index and chromatic number.

Look into changing the algorithm to work with strong as opposed to weak chromatic number. Determine the relationship between Steiner systems and strong colouring.

Given that the chromatic number has been found for given t-designs, determine the effect of different colourings with that identical chromatic number. Instead of stopping the program when the first successful colouring is found, find all such colourings of that chromatic number and then compare them to see what symmetry properties there are. Determine then if there are other orbit representatives and different generators that would generate the same design. Determine if new designs can be generated with results found.

Investigate colouring of partial Steiner systems. Determine if such colouring leads to new designs.

Determine some Steiner quintuple systems and investigate their colouring. Determine if there is a mathematical or symmetry relationship between successful colorings of triple and quadruple and successful colourings of quintuple system.

Adapt the algorithm for coloring to a graph colouring by labelling each vertex of a graph with a unique number. Use blocks of length two to represent edges of the graph in a non-commutative way only, i.e. use increasing vertex-number as you label each vertex of a graph. Then, if an edge is represented by {x,y}, then x is less than y on the unique labelling of the graph edges. Apply the algorithm to this representation of the graph. Since the algorithm now scans looking for minimal

number of uncolored vertices, a step which is not needed when the blocksize is two as it would be in a graph, the algorithm should be simplified for graph colouring to remove this unneeded portion.

Determine if the above representation and coloring lead to a deterministic algorithm for shortest-path traversing of a graph.

Represent Steiner systems, orbit representatives, t-designs on a computer screen graphically using colors to see if such visual representation provides additional information. Use face-color-filling to see visual effect of manipulation and partitioning of the blocks.

Improve the bounds found in [1] by finding the chromatic number of 4-regular 4-uniform hypergraphs.

References

1. N. Alon, Z. Bregman, Every 8-Uniform 8-Regular Hypergraph Is 2-Colorable, *Graphs and Combinatorics,* Vol. 4, (1988), 303-306.
2. C. Berge, *Graphs and Hypergraphs,* North Holland, Amsterdam, (1973).
3. S. Bilaniuk, E. Mendelsohn, A Survey of Colouring of Steiner Systems, *Congressus Numerantium,* Vol. 43, (1984), 127-140.
4. C. J. Cho, A Study in Steiner Quadruple Systems, *M.Sc. Thesis,* McMaster University, Hamilton, Ontario, Canada (1979).
5. C. J. Colbourn, Computing the Chromatic Index of Steiner Triple Systems, *The Computer Journal,* Vol. 25, No. 3 (1982), 338-339.
6. C. J. Colbourn, M. J. Colbourn, The Chromatic Index of Cyclic Steiner 2-Designs, *International Journal Math. and Math Sci.,* Vol. 5, No. 4 (1982), 823-825.
7. C. J. Colbourn, M. J. Colbourn, Greedy Colourings of Steiner Triple Systems, *Annals of Discrete Mathematics,* Vol. 18, (1983), 201-208.
8. C. J. Colbourn, M. J. Colbourn, K.T. Phelps, V. Rödl, Coloring Block Designs is NP-Complete, *SIAM J. Alg.Disc. Meth.,* Vol. 3, No. 3 (1982), 305-307.

9. C. J. Colbourn, M. J. Colbourn, K.T. Phelps, V. Rödl, Coloring Steiner Quadruple Systems *Discrete Applied Mathematics,* Vol. 4, (1982), 103-111.

10. C. Colbourn, J. J. Harms, P. C. van Oorschot, The Role of Combinatorial Designs in Computer Science, (Preprint.)

11. C. J. Colbourn, D. Jungnickel, A.Rosa, The Strong Chromatic Number of Partial Triple Systems, *Discrete Applied Mathematics,* (Preprint.)

12. C. J. Colbourn, W. Kocay, D. Stinson, Some NP-Complete Problems for Hypergraph Degree Sequences, *Discrete Applied Mathematics,* Vol. 14, (1986), 239-254.

13. C. J. Colbourn, A. Rosa, Quadratic Leaves of Maximal Partial Triple Systems, *Graphs and Combinatorics,* (Preprint.)

14. M. J. Colbourn, Algorithmic Aspects of Combinatorial Designs: A Survery, *Annals of Discrete Mathematics.* 26 (1985) 67-136.

15. F. N. Cole, Kirkman Parades, *Bulletin American Mathematical Society,* Vol. 28 (1922), 434-437.

16. M. DeBrandes, K.T. Phelps, Steiner Triple Systems with Small Maximal Independent Sets, *Ars Combinatoria,* Vol. 17 (1984), 15-19.

17. M. DeBrandes, K.T. Phelps, V. Rödl, Coloring Steiner Triple Systems, *SIAM Journal Alg. Discrete Meth.,* Vol. 3 (1982), 241-249.

18. I. Disner, On Cyclic Steiner Systems S(3,4,22), *Annals of Discrete Math,* Vol. 7 (1980), 301-313.

19. J. Doyen, M. Vandensavel, Non-isomorphic Steiner Quadruple Systems, *Bulletin Soc. Math. Belg.,* Vol. 23 (1971), 393-401.

20. A. Ehrenfeucht, V. Faber, H. Kierstead, A New Method of Proving Theorems on Chromatic Index, *Discr. Math.* 52 (1984) 159-164.

21. P. Erdös, On the Combinatorial Problems I Would Most Like to See Solved, *Combinatorica ,* Vol. 1 (1981) 25-42.

22. P. Erdös, *Problems and Results on Chromatic number in finite and infinite graphs,* Graph Theory with Applications to Algorithms and Computer Science, Wiley, New York(1985), 201-213.

23. P. Erdös, Hajnal J., Chromatic Number of Finite Graphs and Hypergraphs, *Discr. Math.* 53 (1985) 281-285.

24. P. Erdös, Hajnal J., *On the Chromatic Number of Graphs and Set Systems,* Acta Math. Acad. Sci. Hungar., 17 (1966) 61-99.

25. P. Erdös, L. Lovasz, *Problems and results on 3-chromatic hypergraphs and related questions,* Infinite and Finite Sets, North-Holland, Amsterdam, 1975.609-627.

26. S. Even, *Algorithmic Combinatorics,* Macmillan, New York, (1973).

27. S. Fiorini, R. J. Wilson, *Edge–Colourings of Graphs.* Pitman, London (1977).

28. M. Hall, *Combinatorial Theory,* Wiley, New York, (1967).

29. F.Harary, J. Maybee, *Graphs and Applications, Proceedings of the 1st Colorado Symposium on Graph Theory,* Wiley, New York(1985).

30. A. Hedayat, S. Kageyama, The Family of t-Designs - Part 1, *J. of Statistical Planning and Inference,* 4 (1980) 173-212.

31. J. E. Hopcroft, On the Harmonious Coloring of Graphs, *SIAM J. Alg.Disc. Meth.,* Vol. 4 (1983) 306-311.

32. S. Kageyama, A. Hedayat, The Family of t-Designs - Part II, *J. of Statistical Planning and Inference,* 7 (1983) 257-287.

33. E.S. Kramer, S.S. Magliveras, R. Mathon, The Steiner Systems S(2,4,25) with Nontrivial Automorphism Group, *(Preprint.)*

34. E.S. Kramer, S.S. Magliveras, V.D. Tonchev, On the Steiner Systems S(2,4,25) invariant under a group of order 9, *Annals of Discrete Mathematics.* 34 (1987) 307-314.

35. E.S. Kramer and D. Mesner, t-designs on Hypergraphs, *Discr. Math.* 15 (1976) 263-296.

36. D.L. Kreher and S.P. Radziszowski, Finding Simple t-Designs by Using Basis Reduction, *Congressus Numerantium, Proceedings of the 17-th Southeastern Conference on Combinatorics, Graph Theory and Computing,* 55 (1986) 235-244.

37. D. Leven, Z. Galil, NP-Completeness of finding Chromatic Index of Regular Graphs, *J. of Algorithms,* 4 (1983) 35-44.

38. C. Lindner, A. Rosa, Steiner Quadruple Systems - A Survey, *Discr. Math.* 22 (1978) 147-181.

39. L. Lovasz, Coverings and Colorings of hypergraphs, *Proceedings of the 4th Southeastern Conference on Combinatorics, Graph Theory and Computing,* (1973) 3-12.

40. F. Macwilliams, N. Sloane, *The Theory of Error-Correcting Code.* North-Holland, Amsterdam (1977).

41. K.Mehlhorn, *Graph Algorithms and NP-Completeness* Springer-Verlag, Berlin (1984).

42. W. Meyer, Equitable Coloring, *American Math. Monthly,* Vol. 80 (1973) 920-922.

43. J. Nesetril, K. T. Phelps, V. Rödl, On the Achromatic Number of Simple Hypergraphs, *Ars Conbinatoria,* Vol. 16 (1983) 95-102.

44. K. Phelps. A Construction of Disjoint Quadruple Systems, *Combinatorial Structures and their Applications, Proceedings of the 8th Southeastern Conference on Combinatorics, Graph Theory and Computing,* (1976) 559-567.

45. K. Phelps. Infinite classes of Cyclic Steiner Quadruple Systems, (Preprint.)

46. K. Phelps, V. Rödl, On the Algorithmic Complexity of Coloring Simple Hypergraphs and Steiner Triple Systems, *Combinatorica,* 4 (1984) 79-88.

47. K. Phelps, V. Rödl, Steiner Triple System with Minimum Independence Number, *Ars Combinatoria,* Vol. Vol. 21 (1986) 167-172.

48. K.Phelps, A. Rosa, 2-chromatic Steiner quadruple systems, *European J. Combinatorics, to appear.*

49. D. Raghavaro, *Constructions and Combinatorial Problems in Design of Experiments.* Wiley, New York (1971).

50. A. Rosa, Colouring Problems in Combinatorial Designs, *Congressus Numerantium, Proceedings of the 16-th Conference Winnipeg/Manitoba 1986, Numerical Mathematics and Computing,* 56 (1987) 45-52.

51. A. Rosa, On the Chromatic Number of Steiner Triple Systems, *Combinatorial Structures and their Applications, Proceedings of the Conference Calgary 1969,* Gorden and Breach, New York (1970).

52. A. Rosa, Steiner Triple Systems and Their Chromatic Number, *Acta Fac. Rerum. Natur. Univ. Comenian. Math.,* 24 (1970) 159-174.

53. A. Satyanarayana, R. Tindell, Chromatic Polynomials and Network Relaibility, *Discr. Math.* 67 (Oct, 1987) 57-79.

54. J. Schmidt, Probablistic Analysis of Strong Hypergraph Coloring Algorithms and Strong Chromatic Number, *Discr. Math.* 66 (Sept 1987) 259-277.

55. J. Schmidt-Pruzan, E. Shamir, E. Upfal, Random Hypergraph Coloring Algorithms and the Weak Chromatic Number, *Journal of Graph Theory,* Vol. 8 (1985) 347-362.

56. M. J. Sharry, A. P. Street, Partitioning sets of quadruples into designs, *(Preprint.)*

57. M. R. Garey, D.S. Johnson, *Computers and Intractability A Guide to the Theory of NP-Completeness,* W.H.Freeman, New York, (1979). 191.