

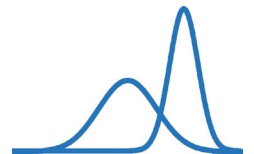
Developing and applying advanced research tools for human perceptual measurement

IFPress® Research Papers

Number 1001

Minimizing Clique Assignments

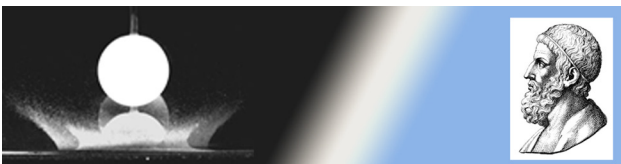
John M. Ennis, Charles M. Fayle and Daniel M. Ennis



The Institute for Perception

7629 Hull Street Rd.
Richmond, VA 23235

Available at www.ifpress.com



Minimizing Clique Assignments

John M. Ennis^{*,a}, Charles M. Fayle^a, Daniel M. Ennis^a

^a*The Institute for Perception, Richmond, VA 23235*

Abstract

The search for minimal clique coverings of graphs appears in many practical guises and with several possible meanings of the term minimal. One reasonable goal is to minimize the number of overall cliques needed for a set of cliques to covering a graph while a second less well-studied but equally reasonable goal is to minimize the number of individual assignments of vertices to cliques needed in order to achieve a covering. Both goals constitute NP-hard problems and as such require efficient algorithms for practical progress to be made towards their resolutions. In this paper we present an efficient technique for accomplishing the latter goal of minimizing the number of clique assignments using a combination of preprocessing techniques and a backtracking algorithm. We then demonstrate that it is not always possible to minimize simultaneously both the number of cliques used and the number of individual clique assignments made with a single clique covering, thereby resolving an open question and underscoring the need for techniques that specifically minimize the number of individual clique assignments made. We then illustrate our approach in two practical examples and report timing results for simulations involving graphs likely to arise in applied statistics, a category of applications for which minimizing individual clique assignments can be particularly important.

Key words:

Backtracking Algorithm, Independence System, NP-hard Problem, Clique Assignment, Clique Covering, Applied Statistics, Compact Letter Displays, Multiple Statistical Comparisons

*Corresponding author

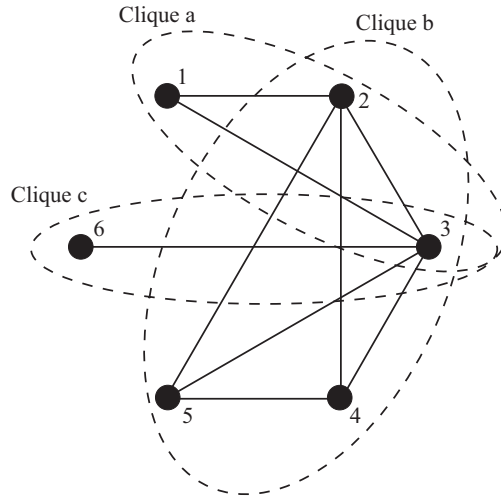


Figure 1: Clique cover example

1. Introduction

A clique within a simple undirected graph is a fully interconnected subgraph. The problem of finding collections of cliques that together cover all vertices and edges of a graph is a well studied problem that appears in many guises [18, 11] and has applications in a wide variety of fields ranging from applied statistics to theoretical biology [1, 24, 19, 23, 15]. To be clear we have the following definitions.

Definition 1.1 (Coverings and Subcoverings). *Suppose \mathcal{C} is a collection of cliques in a graph G . Then \mathcal{C} **covers** G , and \mathcal{C} is called a **covering**, provided every vertex of G appears in at least one clique in \mathcal{C} and every edge of G is also an edge of some clique in \mathcal{C} . If \mathcal{C} is a covering of G and \mathcal{C}' is a second covering of G that can be obtained from \mathcal{C} by removing vertices from cliques in \mathcal{C} then \mathcal{C}' is called a **subcovering** of \mathcal{C} .*

For example, Figure 1 shows a particular clique covering of a graph with six vertices. Finding clique coverings is an NP-hard problem in general and much progress has been made in producing efficient algorithms to find such coverings [18, 4, 2, 6, 14]. Once clique coverings have been found, a natural question is to what degree are these coverings minimal. Complicating matters is the fact that minimality can take on several possible meanings. One

definition of minimal is that the covering contains as few cliques as possible. This definition has been well studied, in particular using data reduction techniques [16], and efficient algorithms have been found [23, 13, 15, 14]. A less well studied problem, and one for which no efficient algorithms are known to produce exact answers, is that of minimizing the number of individual assignments of vertices in the clique covering. This problem has applications in applied statistics, for example, wherein one might wish to minimize the size of a display used to represent the results of a large set of statistical multiple comparisons [22, 23, 15]. In order to distinguish between these two problems we have two definitions of minimal coverings.

Definition 1.2 (Minimal Coverings). *If \mathcal{C} is a clique covering of G and the number of cliques in \mathcal{C} is minimal among all coverings of G then we say that \mathcal{C} is **clique minimal**. If \mathcal{C} is a clique covering of G and the number of clique assignments made by \mathcal{C} is minimal among all coverings of G then we say that \mathcal{C} is **assignment minimal**.*

Assignment minimal coverings are minimal with respect to the number of clique assignments they make. Assignment minimal coverings have not been well studied, but given the progress that has been made [22, 23, 15] in studying clique minimal coverings two natural questions arise. The first is whether assignment minimal coverings are necessarily clique minimal and the second is whether or not it is sufficient to search only among clique minimal coverings when searching for assignment minimal coverings. Later in this paper we show that the answer to both of these questions is no. It is worth noting that both of these questions had been open before the techniques of this paper were applied. Before resolving those questions though we provide an efficient algorithm for finding assignment minimal coverings, the first of its kind. We then show through example that it is not sufficient to examine clique minimal coverings when searching for assignment minimal coverings, thus resolving both of the open questions above and motivating the need for algorithms that produce assignment minimal coverings. Following our example we demonstrate the use of our algorithm in the field of applied statistics and we conclude by reporting simulated timing results for examples such as are likely to arise in practice. We begin by proving some basic results.

	a	b	c
1	1	0	0
2	1	1	0
3	1	1	1
4	0	1	0
5	0	1	0
6	0	0	1

Table 1: Clique membership matrix for covering in Figure 1

2. Basic Results

We begin by introducing the concept of a clique membership matrix¹ of a clique covering. Clique membership matrices will allow us to conduct the computations that follow more simply.

Definition 2.1 (Clique Membership Matrix). *Suppose $\mathcal{C} = \{C_1, C_2, \dots, C_L\}$ is a collection of cliques on the vertices $V = \{v_1, v_2, \dots, v_K\}$. Then the $(K \times L)$ matrix $M = [m_{ij}]$ is the **clique membership matrix** of \mathcal{C} provided $m_{ij} = 1$ when $v_i \in C_j$ and $m_{ij} = 0$ otherwise.*

Table 1 shows a clique membership matrix for the clique covering shown in Figure 1. Using clique membership matrices we can quantify two ways that a clique covering can be considered minimal.

Next we use clique membership matrices to provide an oracle that determines whether a collection of cliques covers a given graph.

Lemma 2.2 (Covering Oracle). *A collection \mathcal{C} of cliques covers a graph G exactly when the clique membership matrix M of \mathcal{C} satisfies*

$$M *_B M^T = A, \tag{1}$$

where $*_B$ denotes Boolean matrix multiplication² and A is the adjacency matrix³ of G .

¹Clique membership matrices depend on the particular order of the cliques and vertices involved. Since this fact is immaterial in the results that follow assuming that one is consistent in one's choice of order, we speak of clique membership matrices of coverings as if they were unique.

²Matrix multiplication according to the rule $1 + 1 = 1$.

³Note that for bookkeeping purposes the vertices of G should be in the same order

Proof. If \mathcal{C} covers G then any two adjacent vertices of G necessarily appear in at least one common clique of \mathcal{C} . Thus the corresponding rows of M have unit entries in at least one common column, making their Boolean inner product nonzero. On the other hand rows corresponding to nonadjacent vertices will not have unit entries in common columns, making their Boolean inner product equal to zero. Equation (1) follows and the converse follows similarly. \square

Although Lemma 2.2 may seem somewhat trivial an essential point is that Boolean matrix multiplication is extremely compatible with bit-wise machine operations, allowing our upcoming techniques to be implemented at a very basic level computationally. Continuing along that path, a special type of clique covering that we will make repeated use of is the so-called maximal covering.

Definition 2.3 (Maximal Covering). *Suppose G is a graph. The covering that consists of the complete set of maximal cliques in G is called the **maximal covering**.*

We will use the maximal covering as a starting point in our search for assignment minimal coverings, according to the following lemma.

Lemma 2.4. *Among the subcoverings of the maximal covering of a graph G there is at least one assignment minimal covering of G .*

Proof. Suppose that \mathcal{C}_1 is a clique covering of G and let \mathcal{M} be the maximal covering. Since each clique in \mathcal{C}_1 is contained in at least one maximal clique we have a possibly non-injective map $f : \mathcal{C}_1 \rightarrow \mathcal{M}$ that sends each clique in \mathcal{C}_1 to one of the maximal cliques that contains it. Set

$$\mathcal{C}_2 = \{\cup f^{-1}(M)\}_{M \in \mathcal{M}}$$

and observe that \mathcal{C}_2 is a subcovering of \mathcal{M} that makes at most as many clique assignments as \mathcal{C}_1 . Thus if \mathcal{C}_1 is assignment minimal then \mathcal{C}_2 is assignment minimal and the result follows. \square

in both M and A and that we adopt the convention that adjacency matrices have unit diagonal. Lemma 2.2 could be restated without the use of this convention but it would be less straightforward to state. Also note that we assume that the clique membership matrix of \mathcal{C} has one row for every vertex in G .

We can further restrict the attention of our search by noting that whenever two vertices of a graph belong to identical cliques we need only consider subcoverings for which those vertices also belong to identical cliques.

Lemma 2.5. *For any clique covering \mathcal{C}_1 of a graph G there exists a second clique covering \mathcal{C}_2 of G with the following properties*

1. \mathcal{C}_2 makes at most as many clique assignments as \mathcal{C}_1
2. Vertices that belong to identical cliques in G belong to identical cliques in \mathcal{C}_2

Proof. Suppose that \mathcal{C}_1 is clique covering of G and that W is a maximal collection of vertices of G that all belong to identical cliques in G . Each vertex in W is involved in a series of clique assignments in \mathcal{C}_1 and some vertex w is involved in the fewest number of such assignments. Since the vertices in W all involve identical connectivity information we can thus create a new clique covering from \mathcal{C}_1 by replacing the clique assignments of every element of W with the clique assignments of w . By construction, this new covering makes at most as many clique assignments as \mathcal{C}_1 . Repeating this process for every maximal set of identically connected vertices we obtain the result. \square

Lemma 2.5 greatly assists us in our search for assignment minimal coverings as it allows us to identify vertices with identical connectivity information as long as we keep track of the number of vertices that have been identified in each case⁴. In other words Lemma 2.5 allows us to effectively preprocess our data and reduce our problem size before we begin our search for assignment minimal coverings. The search itself will be the subject of the next section.

3. Backtracking for Assignment Minimal Coverings

We now present a backtracking⁵ algorithm called FIND-ESS that we use to find assignment minimal coverings of a given graph G . To aid our presentation we use the following terms.

⁴An important point is that if two vertices are identified then clique assignments involving that vertex need to count twice for the purposes of future computations.

⁵Backtracking techniques have historically been extremely valuable in providing practical solutions to NP-hard problems [12, 4, 3, 28, 29].

Definition 3.1 (Removable and Essential Clique Assignments). *If \mathcal{C} is a clique covering of a graph G and a particular vertex can be removed from a particular clique in \mathcal{C} to yield a second covering of G then that particular clique assignment is called **removable**. An assignment that is not removable is called **essential**. If \mathcal{C} makes only essential assignments then the covering \mathcal{C} is called **essential**.*

Note that if a removable assignment is removed to create a new covering then assignments that were previously removable might become essential. FIND-ESS finds all maximal combinations of clique assignments that can be simultaneously removed from a given clique covering of a graph to obtain all essential subcoverings. Thus by Lemma 2.4, applying FIND-ESS to the maximal covering will produce at least one assignment minimal covering.

The details of the algorithm are given in Algorithm 1, which is similar in structure to the classic clique finding algorithm of Bron and Kerbosch [4]. Note that although more advanced clique finding algorithms exist [19, 5, 26, 6, 7] they are not easily adaptable to our purposes as we effectively seek to find all maximal independent sets in an independence system [27, 20, 8]. The algorithm call is FIND-ESS(\emptyset, L, \emptyset), where L is the set of removable letter assignments in the maximal covering. Prior to calling FIND-ESS we initialize a matrix M to be the clique membership matrix of the maximal display. M is globally defined and will change as the algorithm progresses. We also initialize $R(\ell)$ and $C(\ell)$ to be the row and column locations of each clique assignment ℓ in the starting display. For each clique assignment ℓ we then let I_ℓ be the set of column indices corresponding to nonzero entries in row $R(\ell)$ of the adjacency matrix that the starting display represents.

For FIND-ESS to be effective, FIND-ESS must be able to determine quickly whether or not a given letter assignment is removable from the current clique membership matrix M . By Lemma 2.2 we need only check whether

$$M *_B M^T = A \tag{2}$$

continues to hold, where A is the adjacency matrix of G . To see how this can be accomplished quickly, suppose that we attempt to remove a clique assignment ℓ from display, which is equivalent to setting $M(R(\ell), C(\ell))$ equal to zero. In order to verify that Equation 2 continues to hold we need only check the effect of this change on the elements in row $R(\ell)$ of $M *_B M^T$. In fact we only need to compute the Boolean inner products between row $R(\ell)$ and the rows with indices in I_ℓ and verify that these inner products have remained nonzero. If any of these inner products equals

Algorithm 1: FIND-ESS(S, P, X)

```
1 global  $M$ 
2 if ( $P = \emptyset$ ) and ( $X = \emptyset$ ) then
3   report essential covering
4 else
5   assume  $P = \{p_1, \dots, p_K\}$ 
6   for  $k = 1$  to  $K$  do
7      $P_{new} = \emptyset$ 
8      $S_{new} = S \cup \{p_k\}$ 
9     remove  $p_k$  by setting  $M(R(p_k), C(p_k)) = 0$ 
10    for  $j > k$  do
11      if ( $p_j$  is removable) then  $P_{new} = P_{new} \cup \{p_j\}$ 
12     $X_{new} = \emptyset$ 
13    assume  $X = \{x_1, \dots, x_L\}$ 
14    for  $m = 1$  to  $L$  do
15      if ( $x_m$  is removable) then  $X_{new} = X_{new} \cup \{x_m\}$ 
16    FIND-ESS( $S_{new}, P_{new}, X_{new}$ )
17    replace  $p_k$  by setting  $M(R(p_k), C(p_k)) = 1$ 
18     $X = X \cup \{p_k\}$ 
```

	a	b	c	d	e	f	g
1	1	1	1	0	0	0	0
2	1	1	0	1	0	1	0
3	1	1	0	1	0	1	0
4	0	0	1	0	1	0	1
5	1	0	0	1	0	0	0
6	0	0	0	1	1	0	0
7	0	1	1	0	0	1	1
8	0	0	0	1	1	1	1

Table 2: Clique membership matrix for maximal covering where clique minimal is not assignment minimal

zero we know that Equation (2) no longer holds, otherwise we know that the clique assignment ℓ is removable. Since these operations can be implemented at a machine level, FIND-ESS is well suited to high speed computation. In the next section we show via example that in order to find assignment minimal coverings it is not sufficient only to find clique minimal coverings. Thus algorithms such as FIND-ESS that explicitly produce assignment minimal coverings are needed whenever assignment minimal coverings are desired.

4. Clique and Assignment Minimality

Since FIND-ESS effectively conducts a exhaustive search to find all essential coverings, we know at the end of the search how many cliques are in the clique minimal coverings and how many assignments are made by the assignment minimal coverings. Using FIND-ESS we have verified that there are graphs for which no clique minimal covering is assignment minimal. The question as to whether or not this was the case had previously been open. An example of such a graph with eight vertices is given in terms of the clique membership matrix of the maximal display in Table 2. For reference we have labeled the cliques of the maximal covering with the letters "a" through "g." This is a very clean example as there is exactly one clique minimal covering, exactly one assignment minimal covering, and they are different.

To obtain the clique minimal covering one can verify that the cliques "b" and "g" can be removed altogether and that removing these cliques leads

to the removal of seven clique assignments. Once cliques "b" and "g" have been eliminated, however, all the other clique assignments are essential. On the other hand, there is a combination of eight clique assignments spanning several cliques that can simultaneously be removed from the maximal covering, but the removal of these letter assignments only causes the elimination of a single clique. Our searches have shown that is not completely uncommon, meaning that if one wishes to minimize the total number of clique assignments used in a covering one needs to consider more than clique minimal coverings. In the next section we consider an application for which it can be particularly important to minimize the number of individual clique assignments and not just the number of cliques.

5. Application to Applied Statistics

In applied statistics one often seeks to summarize the results of a large number of multiple comparisons in a concise manner. These comparisons could be tests for differences or could involve other pairwise testing such as tests for equivalence [10]. Traditionally underlining or underscoring has been used for this purpose [9, 25] and more recently the results of underlining have been represented using line displays [23, 15]. For an example suppose that five treatments were tested against one another and that the treatment pairs shown in Table 3 were found to be not significantly different from each other. This information can be summarized in a line display as shown in Figure 2.

Treatment A	Treatment B
1	2
1	3
2	3
2	4
3	4
3	5
4	5

Table 3: Non-significantly different treatment pairs

As the number of treatments grows line displays become increasingly valuable for their ability to summarize what might otherwise become an



Figure 2: Line display corresponding to Table 3

overwhelming amount of information. Nonetheless, line displays are limited in their applicability as they assume treatments lying between two non-significantly different treatments will not be significantly different from each other. This assumption holds in many cases, such as when the variances of treatment means are equal, but there exist collections of comparisons for which no line display is applicable [17, 22].

To increase flexibility of representation, Piepho has recommended that letter displays be used instead of line displays [23]. A letter display is similar to a line display in that it also reflects the results of a collection of multiple comparisons. As with line displays, each row of a letter display corresponds to a treatment. If two treatments are found to be not significantly different from each other they are given at least one letter in common. Treatments found to be significantly different from each other are given no letters in common. For example, Table 4 shows a letter display corresponding to Table 3. As one can verify by inspection Table 4 corresponds directly to the line display in Figure 2. Each line display has exactly one such corresponding letter display, up to a relabeling of the letters. More generally, for every collection of multiple comparisons there always exists at least one letter display. Significantly, these letter displays need not be unique even after a relabeling of the letters. For instance, the information displayed in Table 4 is displayed more efficiently in Table 5. Importantly this example

Treatment	Lettering
1	a
2	ab
3	abc
4	bc
5	c

Table 4: A letter display representing Table 3

Treatment	Lettering
1	a
2	ab
3	ac
4	bc
5	c

Table 5: A more efficient letter display representing Table 3

shows that letter displays based on line displays may involve unnecessary letter assignments.

An important connection is that if each letter is viewed as a clique there is a clear correspondence between letter displays and clique membership matrices of clique coverings. Since different letter displays can convey the same information, a natural question is whether a particular letter display makes optimal use of letters. This question is equivalent to asking whether or not a clique covering makes optimal use of its cliques and clique assignments. In [22] and [23] Piepho developed heuristics for letter display reduction, while Gramm *et al.* applied techniques from the study of clique coverings to the problem of finding optimized displays [13, 15, 14]. In particular, Gramm *et al.* have developed exact methods for minimizing the number of distinct letters used in displays, a problem is equivalent to that of minimizing the number of cliques in a covering. Until the methods of this present paper were employed it was not known whether, for a given set of comparisons, letter displays that minimized the number of individual letter assignments made also minimized the number of distinct letters used. The example of the previous section shows that this is not the case, meaning that in order

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	1	0	0
3	1	1	1	1	1	1	1	1	1	1	1	0	0
4	0	1	1	1	1	1	1	1	1	1	1	0	0
5	0	1	1	1	1	1	1	1	1	1	1	0	0
6	0	1	1	1	1	1	1	1	1	1	1	0	0
7	0	1	1	1	1	1	1	1	1	1	1	0	0
8	0	1	1	1	1	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1	1	1	1	1	0
10	0	1	1	1	1	1	1	1	1	1	1	1	1
11	0	1	1	1	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	1	1	1	1	1	1
13	0	0	0	0	0	0	0	1	0	1	1	1	1

Table 6: Nonsignificant differences for a triticale yield experiment

to minimize the use of individual letter assignments one needs to do more than minimize the number of distinct letters used.

We applied our techniques to a pair of examples from the literature. For our first example we considered results from a large scale, multiple location triticale yield experiment. As reported by Piepho [22] we have the matrix of nonsignificant differences shown in Table 6.

Viewing the matrix in Table 6 as an adjacency matrix we used the method of Cazals and Karande [5] to obtain the complete set of maximal cliques. We then trimmed all simultaneously removable cliques assignments from this covering to obtain all essential coverings. In particular we obtained the display given in Table 7, which is minimal in its use of both distinct letters and individual letter assignments. The execution of the algorithm was effectively instantaneous.

For our second example we focused on a large multi-environment wheat yield trial performed by CIMMYT (Centro Internacional de Maiz y Trigo, Mexico). The matrix of nonsignificant differences reported by Piepho [23] is given in Table 8.

Using the above methods we found all essential subcoverings of the appropriate graph. In particular we obtained the display given in Table 9 that is minimal in both its use of distinct letters and number of individual

Treatment	Lettering
1	a
2	ab
3	ab
4	b
5	b
6	b
7	b
8	bc
9	bd
10	bc
11	bc
12	cd
13	c

Table 7: An optimal display for a triticale yield experiment

letter assignments made. There are two important observations to be made regarding this example. The first observation is that there is much repetition in the rows of the adjacency matrix. This repetition translates into repetition in the maximal covering and makes the preprocessing according to Lemma 2.5 very effective. Without preprocessing the search tree in this example contained 46656 nodes. With preprocessing there were 6 nodes in the search tree. Thus preprocessing the data resulted in a roughly 8000-fold improvement in performance.

The second point to be made is that the number of letter assignments and the number of distinct letters used in Table 9 are the same as were reported by Piepho in [23]. An important difference, however, is that the display was not known to be optimal when reported by Piepho.

6. Timing Considerations in Practice

We now use a number of simulations to evaluate the performance of our techniques in practice. Since the number of maximal cliques is bounded above by $3^{\frac{n}{3}}$ [21, 26] it is not practical to consider the maximal covering for arbitrary graphs. Thus for the purposes of these simulations we restrict attention to graphs that are likely to arise in the examination of multiple

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
5	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
7	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
10	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0
11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
12	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1	0	1	0	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20	1	0	0	0	0	0	0	1	0	0	0	1	0	1	1	0	1	0	1	1

Table 8: Nonsignificant differences for a wheat yield experiment

Treatment	Lettering
1	bcd
2	bc
3	ab
4	bc
5	ac
6	bc
7	ac
8	bcd
9	bc
10	ac
11	bc
12	bd
13	bc
14	bcd
15	bcd
16	ac
17	bcd
18	c
19	bcd
20	d

Table 9: An optimal display for a wheat yield experiment

statistical comparisons, which we have already seen is an application in need of algorithms to determine assignment minimal coverings.

Graphs corresponding to multiple comparisons tend to be fairly regular in practice. Once treatments are placed in order by size their corresponding vertices tend to be connected to nearby vertices and disconnected from vertices further away. Only infrequently will connections with further away vertices be included when connections with nearby vertices are omitted. All adjacency matrices generated for our simulations were created according to these guidelines.

For our simulations we used parameters N , p and c . N is the number of treatments being compared, p is the likelihood that two consecutive treatments will not be significantly different from each other and c is a culling parameter that we use to simulate examples that cannot be represented by line displays. For given values of N , p and c we use an algorithm GEN-MAT to simulate adjacency matrices. The details of GEN-MAT are given in Algorithm 2. When $c = 0$, GEN-MAT generates adjacency matrices that can be described by line displays, such as would occur when sample sizes for treatment means are equal. Since there exist collections of multiple comparisons that cannot be represented by line displays [17, 22], we proceed element-wise through the adjacency matrix and determine, for each non-diagonal element, with probability c whether or not to set that element equal to zero. Thus a nonzero value for c allows for occasional gaps in the adjacency matrix. Sample output from GEN-MAT is shown in Table 10.

Using GEN-MAT we simulated 1000 adjacency matrices for a variety of triplets (N, p, c) using a Hewlett-Packard dv7 Notebook PC with an AMD Turion™X2 Dual-Core Mobile RM-72 2.10 GHz Processor, with 4.00GB of RAM and running the 64-bit Windows Vista SP1 Operating System. For each adjacency matrix we computed the complete set of maximal cliques using the method of Cazals and Karande [5] then trimmed the maximal covering using FIND-ESS to find all essential subcoverings. In each case we identified a single assignment minimal covering. In the event that the search did not terminate within 30 seconds we returned a essential subcovering that was minimal among all essential subcoverings found to that point first with respect to the total number of clique assignments made and then with respect to the number of distinct cliques it contained. For each triplet in Table 11 we computed the mean time in seconds required for a search (**MT**), the mean number of clique assignments in the maximal display (**MA**), the

Algorithm 2: GEN-MAT(N, p, c)

```
1 stop = 1
2 for col = 1 to  $M$  do
3   if stop <  $N$  then
4     if (stop = col) then stop = stop + 1
5     rnd  $\sim U(0, 1)$  // random number generation
6     temp =  $p$ 
7     while (stop <  $N$ ) and (rnd > 1 - temp) do
8       stop = stop + 1
9       temp =  $p * temp$ 
10    for row = col to stop do
11      rnd  $\sim U(0, 1)$ 
12      if (rnd >  $c$ ) then
13         $A(\text{row}, \text{col}) = 1$ 
14         $A(\text{col}, \text{row}) = 1$ 
15      else
16         $A(\text{row}, \text{col}) = 0$ 
17         $A(\text{col}, \text{row}) = 0$ 
18    for row = stop + 1 to  $N$  do
19       $A(\text{row}, \text{col}) = 0$ 
20       $A(\text{col}, \text{row}) = 0$ 
21 report  $A$ 
```

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	1	1	1	1	1	0	0	0	0	0
2	1	0	1	1	1	1	1	0	0	0	0	0
3	1	1	0	1	1	1	1	1	1	0	0	0
4	1	1	1	0	1	1	1	0	1	1	1	1
5	1	1	1	1	0	1	1	1	1	1	1	1
6	1	1	1	1	1	0	1	1	1	1	1	1
7	1	1	1	1	1	1	0	1	1	1	1	1
8	0	0	1	0	1	1	1	0	1	1	1	1
9	0	0	1	1	1	1	1	1	0	1	1	1
10	0	0	0	1	1	1	1	1	1	0	1	0
11	0	0	0	1	1	1	1	1	1	1	0	1
12	0	0	0	1	1	1	1	1	1	0	1	0

Table 10: Sample output from GEN-MAT(12, 0.75, 0.01)

mean number of clique assignments removed from the maximal covering to obtain the final subcovering (**MR**) and the proportion of searches that completed before the 30 second time limit expired (**PC**).

From these simulations we see that the algorithm typically performs extremely well. Only in the case of 30 highly interconnected vertices with active culling did the algorithm regularly begin to terminate prematurely. Restricting attention to the examples from this case for which the algorithm did terminate prematurely we find that on average there were 184 clique assignments removed out of an average of 278 clique assignments made by the maximal covering, with the returned display being the best out of an average of 11630 essential subcoverings of the maximal covering found so far. Thus, even when the algorithm was forced to terminate early the subcovering returned was of high quality.

7. Conclusion

In this paper we have provided a technique for finding assignment minimal clique coverings. Even though clique minimal coverings have been well studied, we have shown that it is not always possible to find assignment minimal coverings by searching through those that are clique minimal. Thus it is necessary to have specialized algorithms such as the one presented in this

N	p	c	AT	AA	AR	PC
10	0.25	0.00	< 0.001	10.353	0.011	1.000
10	0.25	0.01	< 0.001	10.361	0.018	1.000
10	0.50	0.00	< 0.001	13.466	0.528	1.000
10	0.50	0.01	< 0.001	13.836	0.633	1.000
10	0.75	0.00	< 0.001	15.478	0.990	1.000
10	0.75	0.01	< 0.001	17.060	1.559	1.000
20	0.25	0.00	< 0.001	20.845	0.061	1.000
20	0.25	0.01	< 0.001	20.895	0.072	1.000
20	0.50	0.00	0.140	32.623	3.373	0.997
20	0.50	0.01	1.208	36.362	5.463	0.966
20	0.75	0.00	0.424	47.332	10.948	0.991
20	0.75	0.01	7.597	71.619	28.920	0.769
30	0.25	0.00	< 0.001	31.075	0.067	1.000
30	0.25	0.01	< 0.001	31.213	0.102	1.000
30	0.50	0.00	3.238	59.754	10.090	0.904
30	0.50	0.01	6.107	70.413	17.461	0.811
30	0.75	0.00	6.934	94.651	32.669	0.809
30	0.75	0.01	24.285	236.675	149.580	0.228

Table 11: Simulation results with a 30 second time limit

paper for the purposes of obtaining assignment minimality. We applied our technique to two examples from applied statistics and we showed through a series of simulated timing tests that our technique is efficient enough to be used for at least one large source of problems in practice.

References

- [1] Agarwal, Alon, Aronov, Suri, 1994. Can visibility graphs be represented compactly? *GEOMETRY: Discrete & Computational Geometry* 12.
- [2] Behrisch, Taraz, 2006. Efficiently covering complex networks with cliques of similar vertices. *TCS: Theoretical Computer Science* 355.
- [3] Bitner, J. R., Reingold, E. M., 1975. Backtrack programming techniques. *Commun. ACM* 18 (11), 651–656.
- [4] Bron, C., Kerbosch, J., 1973. Finding all cliques of an undirected graph. *Commun. ACM* 16.
- [5] Cazals, F., Karande, C., Jul. 2005. An algorithm for reporting maximal c -cliques. *Theor. Comput. Sci.* 349, 484–490.
- [6] Cazals, F., Karande, C., 2007. Reporting maximal cliques: New insights. Rapport de recherche, Institut National de Recherche en Informatique et en Automatique.
- [7] Cazals, F., Karande, C., 2008. A note on the problem of reporting maximal cliques. *Theor. Comput. Sci.* 407 (1-3), 564–568.
- [8] de Jager, B., Banens, J., 2001. VISOR: Vast independence system optimization routine. *Algorithmica* 30, 630–644.
- [9] Duncan, D. B., 1955. Multiple range and multiple F-tests. *Biometrics* 11, 1–42.
- [10] Ennis, D. M., Ennis, J. M., (In press). Hypothesis testing for equivalence based on symmetric open intervals. *Commun. Statist.*
- [11] Garey, M., Johnson, D., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York.
- [12] Golomb, S. W., Baumert, L. D., 1965. Backtrack programming. *J. ACM* 12, 516–524.
- [13] Gramm, J., Guo, J., Hüffner, F., Niedermeier, R., 2006. Data reduction, exact, and heuristic algorithms for clique cover. In: *Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments and the Third Workshop on Analytic Algorithmics and Combinatorics*. SIAM.
- [14] Gramm, J., Guo, J., Hüffner, F., Niedermeier, R., 2009. Data reduction and exact algorithms for clique cover. *ACM J. Exp. Algorith.* 13, 2.2–2.15.
- [15] Gramm, J., Guo, J., Hüffner, F., Niedermeier, R., Piepho, H.-P., Schmid, R., 2007. Algorithms for compact letter displays: Comparison and evaluation. *Comput. Statist. & Data Anal.* 52 (2), 725–736.
- [16] Guo, J., Niedermeier, R., 2007. Invitation to data reduction and problem kernelization. *SIGACT News* 38, 31–45.
- [17] Hsu, J., 1996. *Multiple Comparisons. Theory and Methods*. Chapman and Hall.
- [18] Kellerman, E., 1973. Determination of keyword conflict. *IBM Technical Disclosure Bulletin* 16, 544–546.
- [19] Koch, I., 2001. Enumerating all connected maximal common subgraphs in two graphs. *Theor. Comput. Sci.* 250 (1-2), 1–30.

- [20] Lawler, E., Lenstra, J. K., Kan, H. G. R., 1980. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM J. Comput.* 9, 558–565.
- [21] Moon, J. W., Moser, L., 1965. On cliques in graphs. *Israel J. Math.* 3, 23–28.
- [22] Piepho, H.-P., 2000. Multiple treatment comparisons in linear models when the standard error of a difference is not constant. *Biometrical J.* 42 (7), 823–835.
- [23] Piepho, H.-P., 2004. An algorithm for a letter-based representation of all-pairwise comparisons. *J. Comput. Graph. Statist.* 13 (2), 456–466.
- [24] Rajagopalan, S., Vachharajani, M., Malik, S., 2000. Handling irregular ILP within conventional VLIW schedulers using artificial resource constraints. In *Proc. CASES*, 157–164.
- [25] Steel, R., J.H.Torrie, 1980. *Principles and Procedures of Statistics: A Biometrical Approach*, 2nd Edition. McGraw-Hill.
- [26] Tomita, E., Tanaka, A., Takahashi, H., 2006. The worst-case time complexity for generating all maximal cliques. *Theor. Comput. Sci.* 363, 28–42.
- [27] Tsukiyama, S., Ide, M., Ariyoshi, M., Shirakawa, I., 1977. A new algorithm for generating all the maximal independent sets. *SIAM J. Comput.* 6, 505–517.
- [28] van Dongen, M., 2002. A generalization of the backtracking algorithm. Tech. rep., Cork Constraint Computation Centre.
- [29] Yan, J., Zhang, J., 2008. A backtracking search tool for constructing combinatorial test suites. *Journal of Systems and Software* 81 (10), 1681–1693.