Using algebraic properties of minimal idempotents for exhaustive computer generation of association schemes

K. Coolsaet, J. Degraer,
Department of Applied Mathematics and Computer Science,
Ghent University,
Krijgslaan 281–S9, B–9000 Gent, Belgium
Kris.Coolsaet@UGent.be, Jan.Degraer@UGent.be

Submitted: Nov 10, 2007; Accepted: Feb 4, 2008; Published: Feb 11, 2008
Mathematics Subject Classification: 05E30, 05–04

Abstract

During the past few years we have obtained several new computer classification results on association schemes and in particular distance regular and strongly regular graphs. Central to our success is the use of two algebraic constraints based on properties of the minimal idempotents E_i of these association schemes : the fact that they are positive semidefinite and that they have known rank.

Incorporating these constraints into an actual isomorph-free exhaustive generation algorithm turns out to be somewhat complicated in practice. The main problem to be solved is that of numerical inaccuracy: we do not want to discard a potential solution because a value which is close to zero is misinterpreted as being negative (in the first case) or nonzero (in the second).

In this paper we give details on how this can be accomplished and also list some new classification results that have been recently obtained using this technique: the uniqueness of the strongly regular $(126, 50, 13, 24)$ graph and some new examples of antipodal distance regular graphs. We give an explicit description of a new antipodal distance regular 3-cover of K_{14}, with vertices that can be represented as ordered triples of collinear points of the Fano plane.

1 Introduction and overview

Association schemes are combinatorial objects that satisfy very strong regularity conditions and as a consequence of this, have applications in many branches of combinatorial mathematics : in coding theory, design theory, graph theory and group theory, to name but a few.

The regularity properties of an association scheme are parametrized by a set of integers p_{ij}^k which are called the *intersection numbers* of that scheme. A lot of research has been devoted to classifying association schemes in the following sense : given a specific set of intersection numbers, does a corresponding scheme exist, or can we on the other hand prove nonexistence ? If several schemes exist with the same intersection numbers, are they essentially different ? In other words, what can we tell about isomorphism classes of such schemes ?

Quite a bit of work on this subject has already been done and several tables of 'feasible' intersection numbers and related existence information have been published, especially for the better-known special cases of distance regular and strongly regular graphs [2, 3, 10]. During the past few years also the present authors have made several contributions to this subject [5, 6, 7, 8, 9, 11, 12].

In our case most results were obtained by computer. We have developed special purpose programs to tackle several cases for which a full classification did not yet exist. These programs use standard backtracking methods for exhaustive enumeration, in combination with several special purpose techniques to obtain the necessary efficiency.

On many of the techniques and 'tricks' we use in these programs we have already reported elsewhere [8, 9, 11, 12]. In this paper we will describe the use of two constraints (which haven't been discussed in detail before) that are based on algebraic properties of association schemes. They allow us to prune the search tree extensively and turn out to be very powerful.

Both constraints are based on properties of the minimal idempotents E_i associated with a given scheme : the minimal idempotents are always positive semidefinite, and they have a known rank (which can be computed from the intersection numbers). As a consequence, if we generate the association schemes by building their relation matrices 'column by column', we can check whether the corresponding principal submatrix of E_i is positive semidefinite and does not have a rank which is already too large, before proceeding to the next level.

This is however not so straightforward as it may seem : the standard algorithms from numerical algebra for checking positive semidefiniteness and computing the rank of a matrix, suffer from numerical inaccuracy, and we need to take care not to prune a branch of the search tree because a matrix is mistakenly interpreted as not positive semidefinite or its rank is incorrectly estimated.

In Section 2 we give definitions of the mathematical concepts which are used further on, and we list some well-known properties. Section 3 gives a short description of the algorithm for isomorph-free exhaustive generation we have used, in so far as it is relevant to this paper. In Sections 4 and 5 we discuss the algorithms for checking positive semidefiniteness and computing the rank. Finally, Section 6 lists some new classification results we have obtained using this technique. One of the new schemes discovered has a nice geometrical description which we will discuss in detail.

2 Definitions and well-known properties

Let V be a finite set of n vertices. A d-class association scheme Ω on V is an ordered set $\{ \mathrm{R}_0, \mathrm{R}_1, \ldots, \mathrm{R}_d \}$ of relations on V satisfying the axioms listed below. We use the notation $x \, \mathrm{R}_i \, y$ to indicate that $(x, y) \in \mathrm{R}_i$.

1. $\{ \mathrm{R}_0, \mathrm{R}_1, \ldots, \mathrm{R}_d \}$ is a partition of $V \times V$.
2. R_0 is the *identity relation*, i.e., $x \, \mathrm{R}_0 \, y$ if and only if $x = y$, whenever $x, y \in V$.
3. Every relation R_i is *symmetric*, i.e., if $x \, \mathrm{R}_i \, y$ then also $y \, \mathrm{R}_i \, x$, for every $x, y \in V$.
4. Let $x, y \in V$ and let $0 \le i, j, k \le d$ such that $x \, \mathrm{R}_k \, y$. Then the number

$$p_{ij}^k \stackrel{\mathrm{def}}{=} |\{ z \in V : x \, \mathrm{R}_i \, z \text{ and } z \, \mathrm{R}_j \, y \}|$$

only depends on i, j and k.

The numbers p_{ij}^k are called the *intersection numbers* of Ω. Note that $k_i = p_{ii}^0$ denotes the number of vertices y in relation R_i to a fixed vertex x of V, and does not depend on the choice of x. It also follows that $n = k_0 + \ldots + k_d$ is completely determined by the intersection numbers of Ω.

There are several special cases of association schemes that are of independent interest. For example, a *distance regular graph* G of diameter d is a connected graph for which the distance relations form a d-class association scheme. More precisely, a pair of vertices x, y in G satisfies $x \, \mathrm{R}_i \, y$ if and only if $d(x, y) = i$. A *strongly regular graph* is a distance regular graph of diameter 2. Strongly regular graphs are essentially equivalent to 2-class association schemes. Instead of using intersection numbers, it is customary to define strongly regular graphs in terms of their *parameters* (v, k, λ, μ), with $v = n$, $k = k_1$, $\lambda = p_{11}^1$ and $\mu = p_{11}^2$. A strongly regular graph with these parameters is also called a strongly regular (v, k, λ, μ) graph.

(Several mathematical properties of association schemes are relevant to our generation algorithms. For actual proofs of the properties listed in this section, and for further information on the subject of association schemes and distance regular graphs, we refer to [1, 2, 15].)

With every relation R_i of Ω we may associate a 0–1-matrix A_i of size $n \times n$ as follows : rows and columns of A_i are indexed by the elements of V and the entry at position x, y of A_i is defined to be 1 if and only if $x \, \mathrm{R}_i \, y$, and 0 otherwise. In terms of these matrices the defining axioms of a d-class association scheme Ω translate to

$$\sum_{i=0}^{d} A_i = J, \quad A_0 = I, \quad A_i = A_i^T \quad \text{and} \quad A_i A_j = \sum_{k=0}^{d} p_{ij}^k A_k,$$

where I denotes the $n \times n$ identity matrix, J is the all-one matrix of the same size and A^T is the transpose of A.

It follows readily that $A_0, A_1, \ldots, A_d$ form a basis for a $(d+1)$-dimensional commutative algebra $\mathcal{A}$ of symmetric matrices with constant diagonal. This algebra $\mathcal{A}$ was first studied by Bose and Mesner [4] and is therefore called the *Bose-Mesner algebra* of Ω.

It is well known that $\mathcal{A}$ has a basis of so-called *minimal idempotents* $E_0, \ldots, E_d$ (also called *principal* idempotents), satisfying

$$E_0 = \frac{1}{n}J, \quad E_iE_j = \delta_{ij}E_i, \quad \sum_{i=0}^{d} E_i = I, \quad \text{for all } i,j \in \{0,\ldots,d\}.$$

Since ${E_i}^2 = E_i$, it follows easily that each minimal idempotent E_i is *positive semidefinite*. i.e., that $xE_ix^T \geq 0$ for all $x \in \mathbf{R}^{1\times n}$.

Consider the coefficient matrices P and Q that express the relation between the two bases of $\mathcal{A}$ as follows :

$$A_j = \sum_{i=0}^{d} P_{ij}E_i, \qquad E_j = \frac{1}{n}\sum_{i=0}^{d} Q_{ij}A_i.$$

(P and Q are called the *eigenmatrix* and *dual eigenmatrix* of Ω, respectively.)

It can be proved that $PQ = QP = nI$, that P_{ij} is an eigenvalue of A_j, that the columns of E_i span the corresponding eigenspace, and that E_i has rank Q_{0i}. Let $x, y \in V$, then the definition of E_j implies that the (x,y)-th entry of E_j is equal to Q_{kj}/n where k is the unique class to which the pair (x,y) belongs, or equivalently, the unique index such that $x\,\mathrm{R}_k\,y$.

For the purposes of this paper it is important to note that the entries of P and Q can be computed from the intersection numbers p_{ij}^k of Ω. In other words, we can compute P and Q for a given set of intersection numbers without the need for an actual example of a corresponding association scheme.

Two association schemes $\Omega = \{\,\mathrm{R}_0\,,\ldots,\,\mathrm{R}_n\,\}$ on V and $\Omega' = \{\,\mathrm{R}'_0\,,\ldots,\,\mathrm{R}'_n\,\}$ on V' are called *isomorphic* if there exists a bijection $\pi : V \to V'$ such that for every $i \in \{0,\ldots,d\}$ the following property holds

$$x\,\mathrm{R}_i\,y \text{ if and only if } x^\pi\,\mathrm{R}'_i\,y^\pi, \quad \text{for all } x,y \in V.$$

This means that two association schemes Ω and Ω' are isomorphic if and only if the vertices of V and V' can be numbered in such a way that all corresponding matrices A_i are identical for both schemes.

The problem of *classification* of association schemes consists of finding all association schemes that correspond to a given set of intersection numbers, *up to isomorphism*, i.e., to determine all isomorphism classes and for each class indicate exactly one representative. In our case we try to classify association schemes by means of a computer using *isomorph-free* exhaustive backtracking techniques.

3 Isomorph-free exhaustive generation

Our programs represent an association scheme Ω internally as an $n \times n$ *relation matrix* M with rows and columns numbered by the vertices of V. The entry M_{xy} at position x, y of M contains the index i of the class to which the pair (x, y) belongs, i.e., the unique i such that $x \,\mathrm{R}_i\, y$. This matrix is symmetric and has zero diagonal.

The exhaustive generation algorithm initially starts with a matrix M where all non-diagonal entries are still left uninstantiated (i.e., undefined, unknown — we denote an uninstantiated matrix entry by a question mark). Then each upper diagonal entry M_{xy} (and at the same time its symmetric counterpart M_{yx}) is systematically recursively instantiated with each value of the domain $\{1, \ldots, d\}$.

During this recursive process we use several constraints to prune nodes of the search tree, either because it can be inferred that the partially instantiated matrix can never be extended to the relation matrix of an association scheme with the required parameters, or because every possible extension is known to be necessarily isomorphic to a result we have already obtained earlier.

In [11, 12] we have described most of the constraints we use that are of a combinatorial nature. In this paper which shall concentrate on the constraints that were derived from the algebraic properties of Ω. These constraints are more easily described in terms of the matrices M_{E_i}, with $i \in \{0, \ldots, d\}$, where matrix entries are defined as follows:

$$(M_{E_i})_{xy} = \begin{cases} \frac{1}{n} Q_{ki} & \text{if } M_{xy} = k \in \{0, \ldots, d\} \\ ? & \text{if } M_{xy} = ? \end{cases}$$

Essentially M_{E_i} is the minimal idempotent E_i, except that we allow entries to be uninstantiated. For ease of notation we shall henceforth simply write E_i instead of M_{E_i}.

As has already been mentioned in the introduction, we use the following constraints :

Algebraic constraints *Let $i \in \{0, \ldots, d\}$. Then every completely instantiated leading principal submatrix of E_i*

- *must be positive semidefinite, and*
- *must have rank at most equal to Q_{0i}.*

Indeed, any principal submatrix of a positive semidefinite matrix must again be positive semidefinite, and any principal submatrix of a matrix must have a rank which is at most the rank of the original matrix.

We only consider leading principle submatrices for reasons of efficiency, and of those, we only look at the largest one which is fully instantiated, for if that matrix satisfies the constraint, then it is automatically satisfied for the smaller ones.

For isomorph rejection we have used an orderly approach [14, 17] : of all association schemes in the same isomorphism class we only generate the relation matrix M which has the smallest *column order* certificate $\mathcal{C}(M)$, defined to be the tuple

$$\begin{aligned}\mathcal{C}(M) = (&M_{1,2}, M_{1,3}, M_{2,3}, M_{1,4},...,M_{3,4}, M_{1,5} \ldots \\ &\ldots, M_{n-3,n-2}, M_{1,n-1},...,M_{n-2,n-1}, M_{1,n},...,M_{n-1,n})\end{aligned}$$

of length $(n^2 - n)/2$ obtained by concatenating the upper diagonal entries of M in a *column-by-column* order. We order certificates using the standard lexicographical ordering. Note that the certificate for a leading principal submatrix of M is a prefix of $\mathcal{C}(M)$.

Although other authors seem to favour a row-by-row generation order (see for example [17] in the context of tournaments), in our case a column-by-column strategy turns out to yield results faster, because in this way large leading principal submatrices which are fully instantiated turn up earlier during search and hence the algebraic constraints can be invoked higher up in the search tree, pruning larger subtrees. However, this speed gain seems to be only truely effective when combined with other (look-ahead) criteria which sometimes allow the generation to switch to a row-by-row sequence temporarily.

The unique matrix M which has the smallest certificate in its isomorphism class is said to be *in canonical form*. Checking whether M is in canonical form is very time consuming an therefore we use several additional criteria to speed up this check: lexical ordering of the rows of M and clique checking [9, 12].

A more extensive discussion of the techniques we use for isomorph-free exhaustive generation algorithms of association schemes can be found in the PhD thesis of one of the authors [13].

4 Checking positive semidefiniteness

In the introduction we have already pointed out that it is not possible to use the standard numerical algorithms for checking positive semidefiniteness in unaltered form. The main reason is that we must make sure that numerical errors do not invalidate our results. Moreover, for reasons of efficiency, we should take advantage of the fact that we have to apply the same algorithm several times to matrices that only differ in the values of a few entries.

Recall from linear algebra that a real symmetric matrix $A \in \mathbf{R}^{m\times m}$ is *positive semidefinite* if and only if $xAx^T \geq 0$ for every row vector $x \in \mathbf{R}^{1\times m}$ and its transpose $x^T \in \mathbf{R}^{m\times 1}$. If the matrix A is *not* positive semidefinite then we will call any row vector x for which $xAx^T < 0$ a *witness* for A.

4.1 The basic algorithm

The following theorem serves as the basis for the algorithm we have used in all our generation programs.

Theorem 1 *Consider a real symmetric matrix $A \in \mathbf{R}^{m\times m}$, where*

$$A = \begin{pmatrix} \alpha & a \\ a^T & A' \end{pmatrix}$$

with $\alpha \in \mathbf{R}$, $a \in \mathbf{R}^{1\times m-1}$ and $A' \in \mathbf{R}^{m-1\times m-1}$. Then we distinguish between the following cases:

1. *If $\alpha < 0$, then A is not positive semidefinite. Moreover, $x = (1\ 0\ \ldots\ 0) \in \mathbf{R}^{1\times m}$ is a witness for A.*

2. *If $\alpha > 0$, then A is positive semidefinite if and only if*

$$A' - \frac{a^T a}{\alpha}$$

is positive semidefinite. If $y \in \mathbf{R}^{1\times m-1}$ is a witness for $A' - a^T a/\alpha$ then $x = (-ya^T/\alpha\ y)$ is a witness for A.

3. *If $\alpha = 0$, then A is positive semidefinite if and only if A' is positive semidefinite and $a = 0$. If $a \neq 0$, then we may find $y \in \mathbf{R}^{1\times m-1}$ such that $ya^T \geq 0$ and then each vector $x = (\lambda\ y)$ is a witness for A whenever $\lambda < -yA'y^T/2ya^T$. If A' is not positive semidefinite, then every witness y for A' can be extended to a witness $x = (0\ y)$ for A.*

Proof : Let $\lambda \in \mathbf{R}$, $y \in \mathbf{R}^{1\times v-1}$ and set $x = (\lambda\ y)$. We have

$$xAx^T = (\lambda \quad y^T) \begin{pmatrix} \alpha & a \\ a^T & A' \end{pmatrix} \begin{pmatrix} \lambda \\ y \end{pmatrix} = \lambda^2\alpha + 2\lambda ya^T + yA'y^T. \tag{1}$$

We consider the following three different cases:

1. If $\alpha < 0$, then the right hand side of (1) is less than zero for $\lambda > 0$ and $y = 0$. Hence A is not positive semidefinite and $(1\ 0\ \ldots\ 0)$ may serve as a corresponding witness.

2. If $\alpha > 0$, then we may rewrite the right hand side of (1) as

$$xAx^T = \alpha\left(\lambda + \frac{ya^T}{\alpha}\right)^2 + y\left(A' - \frac{a^T a}{\alpha}\right)y^T, \tag{2}$$

using $ya^T = ay^T$. This expression is nonnegative for every x if and only if every y satisfies $y\left(A' - a^T a/\alpha\right)y^T \geq 0$, i.e., if and only if the matrix $A' - a^T a/\alpha \in \mathbf{R}^{m-1\times m-1}$ is positive semidefinite. If this matrix is not positive semidefinite, and y is a corresponding witness, then for $\lambda = -ya^T/\alpha$ the vector $(\lambda\ y)$ provides a witness for A.

3. Finally if $\alpha = 0$, then the right hand side of (1) reduces to

$$xAx^T = 2\lambda ya^T + yA'y^T, \tag{3}$$

which is linear in λ. This expression is nonnegative for all λ if and only if the coefficient $2ya^T$ of λ is zero and the constant term $yA'y^T$ is nonnegative. Hence the matrix A is positive semidefinite if and only if $ya^T = 0$ and $yA'y^T \geq 0$ for all y, or equivalently, if and only if $a = 0$ and the matrix A' is positive semidefinite. As a consequence, if A' is not positive semidefinite and y is a witness for A', then the vector $(0\ y)$ provides a witness for A. Also, if $a \neq 0$, then we may find y such that $ya^T > 0$ and then any λ satisfying $\lambda < -yA'y^T/2ya^T$ will make (3) less than zero. ∎

This theorem can easily be used as the basis for an algorithm which checks whether a given real symmetric matrix $A \in \mathbf{R}^{m\times m}$ is positive semidefinite. As was already explained in Section 3, we intend to use this algorithm to check positive definiteness of (millions of) potential leading principal submatrices of minimal idempotents E_i for assocation schemes with the requested parameters.

For ease of notation we will denote the element on the i-th row and j-th column of a matrix M by $M[i,j]$ (instead of $M_{i,j}$). Submatrices keep the row and column numbering of the matrices they are part of. For example, the rows and columns of the matrices A and A' in Theorem 1 would be numbered from 1 up to m and from 2 up to m respectively.

Using the notations of Theorem 1, we define

$$A^{(2)} \stackrel{\text{def}}{=} \begin{cases} A', & \text{if } \alpha = 0, \\ A' - a^T a/\alpha, & \text{otherwise.} \end{cases}$$

The matrix obtained by applying the same process to $A^{(2)}$ shall be denoted by $A^{(3)}$, and in a similar way we may define $A^{(4)}, A^{(5)}, \ldots, A^{(m)}$. We also write $A^{(1)} = A$. In general, the matrix $A^{(k)}$ is a symmetric $(m-k+1) \times (m-k+1)$ matrix with rows and columns numbered from k up to m. This yields the following recurrence relation, for all $i, j \in \{k+1, \ldots, m\}$:

$$A^{(k+1)}[i,j] = \begin{cases} A^{(k)}[i,j], & \text{if } A^{(k)}[k,k] = 0, \\ A^{(k)}[i,j] - \dfrac{A^{(k)}[i,k]A^{(k)}[k,j]}{A^{(k)}[k,k]}, & \text{otherwise.} \end{cases} \tag{4}$$

Theorem 1 leads to Algorithm 1 which takes a real symmetric $m \times m$ matrix A as input and returns true or false depending on whether A is positive semidefinite or not. Algorithm 1 needs $O(m^3)$ operations in the worst case. Storage requirements are only $O(m^2)$ because $A^{(k+1)}[i,j]$ can be stored in the same place as $A^{(k)}[i,j]$. Also note that every $A^{(k)}$ is symmetric and therefore only about half of each matrix needs to be stored.

4.2 A useful variant of the basic algorithm

Observe that all comparisons in Algorithm 1 are performed on elements $A^{(k)}[i,j]$ with either $k = i$ or $k = j$. For $i \leq j$ define $B[i,j] \stackrel{\text{def}}{=} A^{(i)}[i,j]$ (and hence $B[1,i] = A[1,i]$). We

Algorithm 1 Checks whether A is positive semidefinite.

```
function isPSD(A : matrix) : boolean
for k ← 1 ··· m do
   if A^(k)[k,k] < 0 then
      return false
   else if A^(k)[k,k] = 0 then
      for j ← k+1 ··· m do
         if A^(k)[j,k] ≠ 0 then
            return false
         end if
      end for
   end if
   compute A^(k+1) using (4)
end for
return true
```

may now reformulate (4) as follows, for all $i, j > k$:

$$A^{(k+1)}[i,j] = \begin{cases} A^{(k)}[i,j], & \text{if } B[k,k]=0, \\ A^{(k)}[i,j] - \dfrac{B[k,i]B[k,j]}{B[k,k]}, & \text{otherwise.} \end{cases}$$

and hence, by repeated application for different k,

$$\begin{aligned} A^{(k+1)}[i,j] &= A^{(1)}[i,j] - \frac{B[1,i]B[1,j]}{B[1,1]} - \frac{B[2,i]B[2,j]}{B[2,2]} - \cdots \\ &\quad \cdots - \frac{B[k-1,i]B[k-1,j]}{B[k-1,k-1]} - \frac{B[k,i]B[k,j]}{B[k,k]}, \end{aligned}$$

where all fractions with zero denominator $B[j,j]$ should be regarded as equal to zero. From this we obtain the following recurrence relation for B :

$$\begin{aligned} B[i,j] &= A[i,j] - \frac{B[1,i]B[1,j]}{B[1,1]} - \frac{B[2,i]B[2,j]}{B[2,2]} - \cdots \\ &\quad \cdots - \frac{B[i-2,i]B[i-2,j]}{B[i-2,i-2]} - \frac{B[i-1,i]B[i-1,j]}{B[i-1,i-1]}, \end{aligned} \tag{5}$$

again omitting all terms with a zero denominator. We use this relation in Algorithm 2, which again needs $O(m^3)$ operations and $O(m^2)$ storage.

It follows from (5) that the value of $B[i,j]$ only depends on the values of $A[k,l]$ with $k \leq i$ and $l \leq j$. Hence, if we want to apply Algorithm 2 subsequently to two matrices A and $\bar{A}$ whose entries only differ at positions (k,l) such that $k > i$ or $l > j$, we can reuse the value of $B[i,j]$ (and a fortiori, all values of $B[x,y]$ with $x \leq i$ and $y \leq j$) which was obtained during the call of isPSD(A), while computing isPSD($\bar{A}$).

Algorithm 2 Checks whether A is positive semidefinite.

```
function isPSD(A : matrix) : boolean
for i ← 1 ··· m do
   B[1, i] ← A[1, i]
end for
for k ← 1 ··· m do
   if B[k, k] < 0 then                          ❶
      return false
   else if B[k, k] = 0 then
      for j ← k + 1 ··· m  do
         if B[k, j] ≠ 0 then                    ❷
            return false
         end if
      end for
   else
      for j ← k + 1 ··· m do
         compute B[k + 1, j] using (5)
      end for
   end if
end for
return true
```

Similarly, if A is a leading principal submatrix of $\bar{A}$, then again all values $B[i, j]$ which were computed during the call to isPSD(A) can be reused for $\bar{A}$. (Of course, these values will only have been calculated completely when A turns out to be positive semidefinite, but if this is not the case, we may immediatly conclude that also $\bar{A}$ cannot be positive semidefinite.)

These properties make Algorithm 2 extremely suitable for use with our generation strategy : the column-by-column order for instantiating matrix entries guarantees that subsequent applications to the algorithm will be done for matrices that differ very little (typically only in their last column). Also recall that we only check leading principle submatrices for positive semidefiniteness.

In fact, even when the last column of a leading principle submatrix is not yet fully instantiated, we may already be able to decide that the full submatrix cannot possibly be positive semidefinite. Indeed, consider condition ❶ in Algorithm 2. By (5) we have

$$B[k, k] = A[k, k] - \frac{B[1, k]^2}{B[1, 1]} - \frac{B[2, k]^2}{B[2, 2]} - \cdots - \frac{B[k-1, k]^2}{B[k-1, k-1]}, \tag{6}$$

again omitting all terms with a zero denominator. Note that statement ❶ will only be called in those cases where all denominators $B[i, i]$ are nonnegative, and hence that all except the first term on the right hand side of the expression above are nonpositive.

Hence, if the leading $k-1 \times k-1$ principal submatrix of A is already fully instantiated, and $A[k,k]$ is known (which is always the case in our generation program), then as soon as the values of $A[1,k],\ldots,A[i,k]$ are available, $B[1,k],\ldots,B[i,k]$ can be computed and the right hand side of (6) can already be partially evaluated. If that partial sum turns out to be negative, $B[k,k]$ will surely be negative and we need not wait to decide that A is not positive semidefinite until the remaining entries of A are instantiated.

4.3 Coping with numerical errors

The algorithms above use various operations on real numbers which by their nature can only be implemented using approximate arithmetic and hence give rise to numerical inaccuracies. It is therefore important to be absolutely certain that any numerical errors that occur are sufficiently small as not to invalidate classification results obtained by our generation program.

For this reason we modify Algorithm 2 in the following way : before returning false, we first compute a witness x for A using the properties of Theorem 1. Then we compute xAx^T and verify whether it is indeed negative, or more exactly, whether it is smaller than $-\epsilon$ for a suitably small $\epsilon > 0$. If not, we revoke our decision, and proceed with the algorithm as before. For condition ❶ we assume that $B[k,k] = 0$ and for condition ❷ we assume that $B[k,j] = 0$.

Note that the value of λ in case 2 of Theorem 1 was chosen to make xAx^T as small as possible, and hence values of xAx^T that are close to zero do not occur very frequently in practice. It is also important to observe that possible rounding errors for the value of xAx^T can be estimated very accurately.

We have

$$xAx^T = \sum_{i,j} x_i x_j A[i,j].$$

The relative error on each symbol x_i, x_j and $A[i,j]$ is essentially 2^{-b}, where b denotes the number of bits used in the mantissa for the floating point representation (we use a very conservative $b = 32$ for Java 'double' numbers). The relative error ρ on each term $x_i x_j A[i,j]$ is the sum of the relative errors on each symbol, hence $\rho = 3 \cdot 2^{-b}$. The absolute error on each term is equal to $\rho |x_i||x_j||A[i,j]|$, which is bounded upwards by $\rho x_{\max}^2 A_{\max}$, where $x_{\max} \stackrel{\text{def}}{=} \max |x_i|$ and $A_{\max} \stackrel{\text{def}}{=} \max |A[i,j]|$. The absolute error on xAx^T is the sum of the absolute errors on each term, and hence at most $\epsilon = m^2 \rho x_{\max}^2 A_{\max}$, a value which can be easily computed at runtime.

Of course this strategy means that we sometimes falsely assume that a matrix is positive semidefinite when it is not, but fortunately this is not a problem. Usually when this happens, the search is pruned when the leading principal submatrix of next higher order is checked, and ultimately the combinatorial constraints will always ensure that any matrix which is finally generated satisfies the definition of an association scheme with the requested parameters. A less accurate implementation of Algorithm 2 simply results in a less efficient generation program.

Finally, remark that rounding errors can be avoided alltogether by using multiprecision (algebraic) integer arithmetic. However, as far as we know, the best algorithms for checking whether a $k \times k$ matrix A is positive semidefinite, still need a number of digits that is linear in k. Early experiments have shown that the resulting programs are still markedly slower than the approximate methods discussed here.

Although Java, the programming language we have used to obtain our results, has a fairly efficient library for multiprecision integer arithmetic, the reader might wonder whether using a computer algebra system would not have been to our advantage. It should be observed that the algorithms are called in the 'inner loop' of the generation program, and therefore it would be important that the computer algebra system would be one that compiles to machine language. But even then we predict that using floating point arithmetic would still be faster.

One idea which might be worth considering, is to use floating point calculations to 'guess' whether A is positive definite and then in some cases 'make sure' by using exact arithmetic on the same matrix. This extra check could for instance be restricted to those cases where the floating point value of xAx^T is less than some small positive threshold.

4.4 Further improvements

The witness x constructed when A is not positive semidefinite also sometimes allows us to identify a smaller principal submatrix A'' of A which is also not positive semidefinite. This happens when some of the coordinates x_i of the witness x are zero, as in that case the elements on the ith row and column of A are not involved in the value of xAx^T. More precisely, if A'' denotes the submatrix obtained from A by removing all rows and columns with indices i for which $x_i = 0$, then also A'' is not positive semidefinite, and the vector x'' obtained from x by removing all zero entries may serve as a witness for A''.

This observation leads to the following look-back strategy which we can incorporate into our generation algorithm. Suppose that the instantiation of an entry of A completes the submatrix A'' as described above, then this instantiation remains forbidden until a backtrack occurs to the entry of A'' which was instantiated second to last.

For this technique to be effective, it is necessary to have witnesses that have many entries that are zero. To increase the likelyhood of such a witness, we can sometimes 'improve' the witness generated in the second case of Theorem 1 : when we obtain the witness y for $A' - a^T a/\alpha$, we can simply check whether $(0\ y)$ happens to be a witness for A, i.e., whether $yA'y^T < 0$. If this is the case we use $(0\ y)$ instead of the witness $(\lambda\ y)$ of Theorem 1.

A further improvement is possible when the witness x for A turns out to be of the form $x = (x_1, \ldots, x_k, 0, \ldots, 0, x_m)$ for some $k \in \{1, \ldots, m-2\}$ such that $x_k, x_m \neq 0$. In this case, let $\bar{A}$ denote a matrix of size $\bar{m} \times \bar{m}$ (with $k < \bar{m}$) which has the same leading principal $k \times k$ submatrix as A and such that the first k elements of the last columns of both matrices are the same, i.e., such that $A[1, m] = \bar{A}[1, \bar{m}], \ldots, A[k, m] = \bar{A}[k, \bar{m}]$. Then x is also a witness for $\bar{A}$ and $\bar{A}$ is not positive semidefinite either.

This observation leads to the following constraint recording technique. We store the tuple $(A[1, m], \ldots, A[k, m])$ and use it to forbid similar column prefixes, as long as the entries of the $k \times k$ leading principle submatrix are left unaltered. Because the possible entries for $A[i, j]$ can only take a few different values, it is advantageous to store these column prefixes into a trie, which also allows them to be combined to yield even shorter forbidden column prefixes.

5 Checking the rank

As was pointed out in Section 3 we also need a method for checking whether a square symmetric matrix A has a rank which does not exceed a given value r. As with positive semidefiniteness, the standard algorithm for determining the rank (through Gaussian elimination) suffers from numerical inaccuracy, which is this time an even more dangerous problem, because to compute the rank exactly we need to determine with certainty whether various real numbers are equal to zero or not.

The technique which we have used to surmount these problems, and which we describe below, is not as efficient as the algorithms from Section 4, but luckily it turns out that rank checking is not needed as often as checking for positive definiteness. Observe for instance that the rank of a matrix cannot be larger than its order, and therefore we only need to check matrices of size at least $r+1 \times r+1$. Also, we can sometimes postpone rank checking even longer, because adding a row (or column) to a matrix can only increase the rank by 1, and hence the rank of a $m \times m$ matrix A can be at most $2(m - m')$ larger than the rank of a principal submatrix A' of A of order $m' \times m'$. Indeed, if A' turns out to have rank equal to only $r - e$, then rank checking is superfluous until we reach a submatrix of order larger than $m' + e/2$.

We use modular arithmetic to avoid problems of numerical inaccuracy. Recall from linear algebra that the rank of a matrix A with integral entries is always greater than or equal to its *p-rank*, i.e., the rank of the matrix $\bar{A}$ obtained from A by reducing every matrix entry of A modulo a fixed prime number p. So, if we find that the p-rank of A is larger than r, we know for sure that also the rank of A itself is larger than r. The p-rank can be computed exactly, using Gaussian elimination, and if we choose $p < 2^{15}$ (e.g. $p = 32749$), then this can be done using standard 32-bit integer arithmetic without the risc of overflow.

In our generation programs, the entries of the matrices A for which we need to compute the rank are the entries of the dual eigenmatrix Q associated with the association scheme, divided by n. For most of the actual cases which we have investigated these entries are rational numbers. If we multiply A with the least common multiple of the denominators of these constants, we obtain an integer matrix with the same rank, for which we then compute the p-rank.

In the general case, the entries of A can also be irrational, but they will always be algebraic integers. Also then it is still possible to find a suitable p and to transform A into an appropriate matrix $\overline{A}$. For example, the dual eigenmatrix for the Perkel graph [8] has entries that belong to $\mathbf{Q}(\sqrt{5})$. Multiplying A with the appropriate constant yields a matrix with entries in $\mathbf{Z}[\sqrt{5}]$. Choosing $p = 30011$, we obtain $\bar{A}$ by mapping an entry of the form $x + y\sqrt{5}$ to $x + 6583y \bmod p$. Because $6583^2 = 5 \bmod p$, we again end up with a matrix $\bar{A}$ whose rank cannot be larger than the rank of A.

In each of these cases it is of course possible that the rank of $\bar{A}$ is strictly smaller than that of A, and hence that we fail to recognise a matrix A with a rank that exceeds the given bound r. As with the algorithm of the previous section, this is not a problem. It may make the algorithm less efficient, but will never yield results that are not correct.

The remarks of Section 4.3 also hold for computing the rank : using the p-rank as an approximation appears to be more efficient than computing the exact rank with multiprecision arithmetic. It might however be beneficial to use the following 'hybrid' algorithm: if the p-rank 'almost' reaches the bound r, exact arithmetic could be used to compute the true rank. Although in that case it would probably be faster to simply recompute the p-rank for a few additional values of p.

6 Classification results

In [8, 9, 11, 12] we have already discussed many of the results which we have obtained by applying the techniques of the previous sections. In this section we will list some recent classification results.

As was mentioned before, our programs were written in the Java programming language. Timings given are for a computer with a single CPU with a clock speed of approximately 1GHz. All timings are approximate.

6.1 A unique strongly regular $(126, 50, 13, 24)$ graph

There exists a strongly regular graph with parameters $(v, k, \lambda, \mu) = (126, 50, 13, 24)$ which is related to the Hermitian two-graph $\mathcal{H}(5)$ and can be constructed from the Hoffman-Singleton graph [16].

Our generation program proves that this strongly regular graph is uniquely determined by its parameters (up to isomorphism). This took approximately 5 days of processing time.

6.2 Two distance regular antipodal 3-covers of K_{14}

A distance regular graph on 42 vertices with intersection array $\{13, 8, 1; 1, 4, 13\}$, i.e., with $n_1 = 13, p^1_{12} = 8, p^2_{13} = 1, p^1_{10} = 1, p^2_{11} = 4, p^3_{12} = 13$, is called an *antipodal cover of* K_{14} because the relation of being at maximal distance is an equivalence relation (the graph is *antipodal*) and the corresponding quotient (with vertices the 14 equivalence classes of size 3) can be interpreted as a complete graph on 14 vertices (every vertex in every equivalence class is adjacent to exactly one vertex of every other equivalence class).

There is a well-known construction of a graph with this parameters, based on the finite field GF(13) (see for instance [2, Proposition 12.5.3]). This scheme has automorphism group PGL(2, 13) of size 2184.

Our generation program proves that there are exactly 2 isomorphism classes of association schemes with these parameters (this takes about 20 minutes). The second graph can be constructed as follows :

- Vertices are the 42 ordered triples of collinear points in the Fano plane PG(2, 2).
- Two triples (x_0, x_1, x_2), (y_0, y_1, y_2) that correspond to different lines of the Fano plane are adjacent in the graph if and only if either $x_0 = y_0$, $x_1 = y_2$ or $y_1 = x_2$.
- Two triples that correspond to the same line are adjacent if and only if they share the third coordinate, i.e., if they are of the form (x_0, x_1, x_2) and (x_1, x_0, x_2).

The automorphism group of this graph is PGL(3, 2). The antipodal classes correspond to the oriented lines $\{(x_0, x_1, x_2), (x_1, x_2, x_0), (x_2, x_0, x_1)\}$ of the Fano plane. It is fairly easily seen that every triple is adjacent to exactly one triple for each of the 13 remaining oriented lines, in other words, that the graph is a cover of K_{14}. We do not know of an elegant mathematical proof of the fact that the graph is distance regular with the indicated parameters, but this property is easily checked by computer.

6.3 Four distance regular antipodal 3-covers of K_{17}

A distance regular graph on 51 vertices with intersection array $\{16, 10, 1; 1, 5, 16\}$, is an *antipodal cover of* K_{17}. Our generation program proves that there are 4 isomorphism classes of association schemes with these parameters, with automorphism groups of sizes 16320, 240, 192 and 48 respectively (using about 10 hours of time). The first case again corresponds to the construction of [2, Proposition 12.5.3] with group PΓL(2, 16).

References

[1] Bailey R.A., Association Schemes, Designed Experiments, Algebra and Combinatorics, Cambridge studies in advanced mathematics **84** (2004).

[2] Brouwer A.E., Cohen A.M., Neumaier A., Distance-Regular Graphs, Ergeb. Math. Grenzgeb. (3) **18**, Springer-Verlag, Berlin (1989).

[3] Brouwer A. E., *Strongly Regular Graphs*, Handbook of Combinatorial Designs second edition, Ch. J. Colbourn, J. H. Dinitz (eds.), Chapman & Hall/CRC (2006), 852–868.

[4] Bose R.C., Mesner D.M., *On linear associative algebras corresponding to association schemes of partially balanced designs*, Annals of Mathematical Statistics **47** (1952), 151–184.

[5] Coolsaet K., *Local structure of graphs with* $\lambda = \mu = 2$, $a_2 = 4$, Combinatorica, **15(04)** (1995), 481–488.

[6] Coolsaet K., *A distance regular graph with intersection array* $(21, 16, 8; 1, 4, 14)$ *does not exist*, European Journal of Combinatorics, **26(5)** (2005), 709–716.

[7] Coolsaet K., *The uniqueness of the strongly regular graph* srg$(105, 32, 4, 12)$, Bulletin of the Belgian Mathematical Society — Simon Stevin, **12(5)** (2005), 707–718.

[8] Coolsaet K., Degraer J., *A computer assisted proof of the uniqueness of the Perkel graph*, Designs, Codes and Cryptography **34(2–3)** (2005), 155–171.

[9] Coolsaet K., Degraer J., Spence E., *The Strongly Regular (45,12,3,3) graphs*, Electronic Journal of Combinatorics **13(1)** (2006), R32.

[10] Dam, E.R. van, *Three-class association schemes*, Journal of Algebraic Combinatorics, **10(1)** (1999), 69–107.

[11] Degraer J., Coolsaet K., *Classification of three-class association schemes using backtracking with dynamical variable ordering*, Discrete Mathematics **300(1–3)** (2005), 71–81.

[12] Degraer J., Coolsaet K., *Classification of some strongly regular subgraphs of the McLaughlin graph*, to appear in Discrete Mathematics (2007), doi:10.1016/j.disc.2006.11.055

[13] Degraer J., *Isomorph-free exhaustive generation algorithms for association schemes* (2007), PhD thesis.

[14] Faradžev I.A., *Constructive enumeration of combinatorial objects*, Problèmes Combinatoires et Théorie des Graphes Colloque Internat. CNRS, **260** (1978), 131–135.

[15] Godsil C., Royle G., Algebraic graph theory, Springer-Verlag (2001).

[16] Haemers W.H., Kuijken E., *The Hermitian two-graph and its code*, Linear Algebra and its Applications, **356(1-3)** (2002), 79-93.

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