Decomposing Gram matrices

Richard P. Brent
MSI & CECS
ANU

Describing joint work with
Will Orrick, Judy-anne Osborn
and Paul Zimmermann

14 May 2010

Abbreviations

- maxdet — Hadamard maximal determinant problem
  or a matrix with maximal determinant
- JO — Judy-anne Osborn
- PZ — Paul Zimmermann
- WO — Will Orrick
- OS — Will Orrick and Bruce Solomon
- BOOZ — Brent, Orrick, Osborn & Zimmermann

Abstract

When searching for maximal or large-determinant $\{-1, +1\}$ matrices R we construct putative Gram matrices G and try to decompose them — either find R such that $G = RR^T$ or show that no such decomposition exists. We may also know another matrix H such that (possibly) $H = R^T R$; this extra information can be used to speed up the search for R. In this talk we outline a backtracking search algorithm to find R or prove that it does not exist. The algorithm is similar to one used by Orrick to find a maximal matrix of order 15, but has some significant differences. We describe our C implementation, its current limitations, and mention possible future improvements.

The problem

Given $n \in \mathbb{Z}$, $n > 0$, we want to find matrices $R \in \{\pm 1\}^{n \times n}$ with maximal (or sometimes just large) determinant $\det(R)$.
(Or $|\det(R)|$ since the sign is irrelevant.)

The case $n = 0 \bmod 4$ is the "Hadamard order" case which you have heard about in Jennifer Seberry's talks and will hear more about tomorrow.

If $n > 2$ and $n \neq 0 \bmod 4$, we can not meet Hadamard's upper bound $n^{n/2}$, but we can still ask for a matrix with maximal determinant. There are applications to "D-optimal designs".

We shall restrict attention to the case n odd, since the case $n = 2 \bmod 4$ has some technical differences. When n is odd, we naturally get two sub-cases: $n = 1 \bmod 4$ and $n = 3 \bmod 4$.

Our approach

Our approach is similar to that used in 2004 by WO to find the maxdet matrix of order $n = 15$ (following earlier work by Ehlich, Moyssiadis, Kounias and Chadjipantelis for other orders).

First we find putative *Gram matrices* G as described in PZ's talk this morning; then we try to decompose them

$$G = RR^T, \quad R \in \{\pm 1\}^{n \times n}.$$

Some history of the decomposition program(s)

In her Honours thesis (2002), JO outlined essentially the procedure that we use, and it was implemented (in Mathematica) at about the same time by WO.

In April–May 2009, JO and I visited PZ in Nancy, and a prototype Maple program was written (by PZ, with us looking over his shoulder).

After returning to Canberra, JO and I started writing a C program based on PZ's Maple program. This project soon "forked" to give two programs, one (which I will describe today) aiming for efficiency in the decomposition, and the other (to be mentioned in JO's talk) to give information about the search trees.

More details

We can assume that PZ's Gram-finding program finds, up to equivalence, *all* the matrices G satisfying the obvious necessary conditions for a decomposition to exist, and having $\det(G) \geq B^2$ for some bound $B \leq n^{n/2}$.

The conditions on G include:

- $G \in \{-n, \ldots, +n\}^{n \times n}$, $\operatorname{diag}(G) = nI$;
- G symmetric and positive definite;
- $\det(G)$ a perfect square;
- the elements $g_{i,j}$ of G satisfy $g_{i,j} = n \bmod 4$ (this is not necessary, but we can assume it without loss of generality).

Equivalence

If P and Q are signed permutation matrices, then PRQ is equivalent to R (for our purposes) since $\det(PRQ) = \pm\det(R)$. We write $PRQ \sim R$.

If $G = RR^T$ then $(PRQ)(PRQ)^T = PGP^T$, so we can regard G as equivalent to PGP^T. In other words, we are free to permute rows and columns of G and/or change their signs (but preserving symmetry). We write $PGP^T \simeq G$.

Similarly, we can permute the columns of R, and/or change their signs, since $(RQ)(RQ)^T = RR^T$.

Using pairs of Gram matrices

If $G = RR^T$, then there is a "dual" Gram matrix $H = R^T R$. Also, $\det(H) = \det(G)$, so PZ's program should have found (a matrix equivalent to) H as well as G. Using H as well as G gives more constraints on possible factors R, so speeds up the search.

Since $G = RHR^{-1}$ is similar to H, the matrices G and H have the same characteristic equation. Thus, if PZ's program generates a list of matrices $G_1, \ldots, G_m$ say, we can restrict attention to pairs (G_i, G_j) with the same characteristic equation (and *a fortiori* the same determinant).

Effect of signed permutations on H

If, instead of $H = R^T R$, we have a (symmetric) permutation $\widetilde{H} = Q^T H Q$, then

$$\widetilde{H} = \widetilde{R}^T \widetilde{R}, \quad G = \widetilde{R}\widetilde{R}^T,$$

where $\widetilde{R} = RQ$. Thus, we should obtain a solution $\widetilde{R} \sim R$.

Without loss of generality, we can assume that $H = \widetilde{H}$ and $R = \widetilde{R}$.

Weakly self-dual solutions

If $R^T \sim R$, we say that R is *self-dual*.

If $RR^T \simeq R^T R$, we say that R is *weakly self-dual*.

In all known cases, if R is a maxdet matrix, then R is weakly self-dual. Is this always true?

Note: The conjectured maxdet matrix of order 27 (with determinant $546 \times 6^{11} \times 2^{26}$, found by Tamura) is weakly self-dual but not self-dual. Similarly for some Hadamard matrices of orders ≥ 16.

Linear constraints

Suppose that $G = RR^T$ and

$$R^T = [r_1|r_2|\cdots|r_n],$$

i.e. the rows of R are $r_1^T, \ldots, r_n^T$. Then

$$r_i^T r_j = g_{i,j},\ 1 \le i, j \le n.$$

If we already know the first k rows, then we get k *linear* constraints involving row $k+1$:

$$r_i^T r_{k+1} = g_{i,k+1} \text{ for } 1 \le i \le k.$$

Pruning the search space

Recall that we can permute columns of R without changing $G = RR^T$.

When finding row $k+1$ we can permute columns to obtain the lexicographically least solution, subject to the constraint that rows $1, \ldots, k$ are unchanged.

If we are trying to find a solution that also satisfies $H = R^T R$, then the permutations have to be restricted (with a slight abuse of notation, they have to lie in the automorphism group of H).

Example

For example, writing "$-$" for -1, "$+$" for $+1$, "$|$" to show a block boundary, and taking $n = 7$, we might consider a first row

$$- - -| + + + +$$

then a second row

$$- -| + |- -| + +$$

then a third row

$$-| + |- |- | + |- | + |$$

The blocks form a tree: row k contains at most 2^k blocks, and each block at row k splits into at most two blocks at row $k+1$ (until eventually each block is a singleton and can not be divided further).

Using the block structure

Suppose we have a block of size m at row $k+1$. In general there are 2^m possible ways of filling the block with elements of $\{\pm 1\}$. However, we only need to distinguish $m+1$ ways, corresponding to say x entries $+1$ and $m-x$ entries -1.

Suppose there are m blocks, with corresponding "x" values $x_1, \ldots, x_m$. We can express the k linear constraints as an underdetermined system of k linear equations in the m variables $x_1, \ldots, x_m$. Of course, the x_i have to be nonnegative integers satisfying certain upper bounds (the corresponding block sizes).

Storing the tree

In order to make it easy to construct the linear constraints and to backtrack where necessary, we store the tree in a 3-D array which is called the *part* structure (short for “partition”.)

A more storage-efficient data structure is possible, but storage of order n^3 is not a problem for the values of n that we are considering (typically $n < 40$).

Solving the linear equations

We have k linear equations in $m > k$ variables. The corresponding matrix has full rank (i.e. rank k) because G is positive definite.

Using Gaussian elimination with column pivoting, we can assume that the leading $k \times k$ matrix is nonsingular. This corresponds to k "basic" variables x_i, $i \in \mathcal{B}$.

The remaining $m - k$ "non-basic" variables x_i, $i \in \overline{\mathcal{B}}$ can be regarded as parameters. We enumerate the non-basic variables exhaustively, and obtain the basic variables by a matrix-vector multiplication, since the linear constraints imply that the basic variables are an affine function of the non-basic variables.

Checking constraints

If the basic variables thus obtained are not integral or lie outside their bounds, there is no solution corresponding to the given set of non-basic variables.

Weighting columns

Suppose the bounds on variable x_i are $0 \le x_i \le u_i$. The enumeration of non-basic variables involves

$$U = \prod_{i \in \overline{\mathcal{B}}} (u_i + 1)$$

matrix-vector multiplications. Thus, to minimize U, it is best to have the variables with large upper bounds u_i in the basis.

One way of doing this, without provoking numerical instablility, is to give column i a *weight* proportional to u_i. Then the partial pivoting is automatically biased towards selecting columns with large u_i so they tend to go in the basis.

For example, to decompose the conjectured maxdet Gram matrix of order 29 (determinant $(320 \times 7^{12} \times 2^{28})^2$) took 9.4 seconds without weights but only 0.18 seconds with weights.

Quadratic constraints

How can we take advantage of the constraint $R^T R = H$? One way would be to build up columns of R at the same time as we build rows of R using the constraint $RR^T = G$. It is easier (and probably faster) to build rows of R, but prune the search tree using the information provided by H.

We have the relations

$$G^{q+1} = RH^q R^T, \; q \in \mathbb{Z}$$

(at most n such are linearly independent, by the Cayley-Hamilton theorem). We can use these relations to prune the search when generating R by rows.

It is best to use the relation for $q = 2$ instead of (or as well as) the relation for $q = 1$, so as to use information from the lower right corner of H earlier.

Use of linear programming

We have an *integer* programming problem, but if we drop the constraint that the variables x_i are integers, we get a *linear* programming problem: find a feasible solution to a system of linear equations subject to some inequality constraints.

If some subset of the non-basic variables have (integer) values assigned, then the remaining non-basic and basic variables have to satisfy a (smaller) linear program.

Using linear programming reduces the size U of the search space, but at the cost of making each step slower. We have not found a convincing example where linear programming actually speeds up the search, but it could happen for sufficiently large problems.

Randomised search

In cases where *G is* decomposable but it is difficult to find a factor using a deterministic search, we can often do better with a randomised search.

For example, if $n = 27$, there is a known Gram matrix G which decomposes into RR^T, giving a $\{\pm 1\}$ matrix R of determinant $546 \times 6^{11} \times 2^{26}$ which is conjectured to be maximal [Tamura, 2005].

A randomised search explored $1,295,826$ nodes and found a solution in 8 hours; the deterministic search explored over 100 times as many nodes but only reached depth 17 before the system crashed. The tree size is probably greater than 4×10^9.

Some computational results

Computations were performed on the MSI's AMD cluster *orac*:

- 224 AMD 64-bit processors
  (56$\times$ quad-core 2.3 GhZ Opterons)
- 1 GB RAM each

Order 19

$19 = 3 \bmod 4$ so order 19 is harder than say order 21 (done in 1987) or order 25 (1959).

WO (2004) said: “Convincing conjectures exist for the maximal determinants for orders n = 19, 22, 23, and 37. We hope that the methods of this paper can be extended to handle at least some of these cases. Order 19 may be tractable using the current method with efficiency improvements in the computer code, and perhaps parallelization.”

The prediction was correct, at least regarding orders 19 and 37!

Results for order 19

We took $B = 833 \times 4^6 \times 2^{18}$ (97.5% of the Ehlich bound) since there is a known $\{\pm 1\}$ matrix R_1 with determinant B (found by Smith, 1988, and conjectured to be optimal).

More precisely, there are known Gram matrices G_1, G_2 with determinant B^2, such that

$G_1 = R_1 R_1^T$ [Smith],

$G_2 = R_2 R_2^T = R_3 R_3^T$ [Cohn, 2000; OS, 2003],

where R_1, R_2, R_3 are pairwise inequivalent.

The Gram matrix G_1

The Gram matrix found and decomposed by Smith is

$$G_1 = \begin{pmatrix}
19 & 3 & 3 & 3 & 3 & 3 & 3 & - & - & - & - & - & - & - & - & - & - & - & - \\
3 & 19 & 3 & 3 & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
3 & 3 & 19 & 3 & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
3 & 3 & 3 & 19 & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
3 & - & - & - & 19 & 3 & 3 & - & - & - & - & - & - & - & - & - & - & - & - \\
3 & - & - & - & 3 & 19 & 3 & - & - & - & - & - & - & - & - & - & - & - & - \\
3 & - & - & - & 3 & 3 & 19 & - & - & - & - & - & - & - & - & - & - & - & - \\
- & - & - & - & - & - & - & 19 & 3 & 3 & - & - & - & - & - & - & - & - & - \\
- & - & - & - & - & - & - & 3 & 19 & 3 & - & - & - & - & - & - & - & - & - \\
- & - & - & - & - & - & - & 3 & 3 & 19 & - & - & - & - & - & - & - & - & - \\
- & - & - & - & - & - & - & - & - & - & 19 & 3 & 3 & - & - & - & - & - & - \\
- & - & - & - & - & - & - & - & - & - & 3 & 19 & 3 & - & - & - & - & - & - \\
- & - & - & - & - & - & - & - & - & - & 3 & 3 & 19 & - & - & - & - & - & - \\
- & - & - & - & - & - & - & - & - & - & - & - & - & 19 & 3 & 3 & - & - & - \\
- & - & - & - & - & - & - & - & - & - & - & - & - & 3 & 19 & 3 & - & - & - \\
- & - & - & - & - & - & - & - & - & - & - & - & - & 3 & 3 & 19 & - & - & - \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & 19 & 3 & 3 \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & 3 & 19 & 3 \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & 3 & 3 & 19
\end{pmatrix}$$

Here, as usual, "$-$" means "-1".

The Gram matrix G_2

The Gram matrix decomposed by Cohn and (differently) by OS is:

$$G_2 = \begin{pmatrix}
19 & - & - & 3 & - & - & 3 & - & - & 3 & - & - & - & - & - & - & - & - & - \\
- & 19 & 3 & 3 & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
- & 3 & 19 & 3 & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
3 & 3 & 3 & 19 & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
- & - & - & - & 19 & 3 & 3 & - & - & - & - & - & - & - & - & - & - & - & - \\
- & - & - & - & 3 & 19 & 3 & - & - & - & - & - & - & - & - & - & - & - & - \\
3 & - & - & - & 3 & 3 & 19 & - & - & - & - & - & - & - & - & - & - & - & - \\
- & - & - & - & - & - & - & 19 & 3 & 3 & - & - & - & - & - & - & - & - & - \\
- & - & - & - & - & - & - & 3 & 19 & 3 & - & - & - & - & - & - & - & - & - \\
3 & - & - & - & - & - & - & 3 & 3 & 19 & - & - & - & - & - & - & - & - & - \\
- & - & - & - & - & - & - & - & - & - & 19 & 3 & 3 & - & - & - & - & - & - \\
- & - & - & - & - & - & - & - & - & - & 3 & 19 & 3 & - & - & - & - & - & - \\
- & - & - & - & - & - & - & - & - & - & 3 & 3 & 19 & - & - & - & - & - & - \\
- & - & - & - & - & - & - & - & - & - & - & - & - & 19 & 3 & 3 & - & - & - \\
- & - & - & - & - & - & - & - & - & - & - & - & - & 3 & 19 & 3 & - & - & - \\
- & - & - & - & - & - & - & - & - & - & - & - & - & 3 & 3 & 19 & - & - & - \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & 19 & 3 & 3 \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & 3 & 19 & 3 \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & 3 & 3 & 19
\end{pmatrix}$$

Our results for $n = 19$

PZ's Gram finding program was run on 50 processors of orac. The search was split into 2766 tasks, each corresponding to a node at level 5 of the search tree (so each processor did 55 or 56 tasks).

After 188 hours, all processors had finished their tasks, and a total of 9 putative Gram matrices had been found, including the known matrices G_1 and G_2. The total CPU time was about 900 hours (so the load was not well-balanced).

In fact, all 9 candidates were found after 4 hours — the remaining 184 hours were spent in a fruitless search for more. This seems to be typical behaviour (e.g. for $n = 29$, $B = 329 \times 7^{12} \times 2^{28}$).

Details of the cases

The square roots of the determinants of the putative Gram matrices G, divided by $4^6 \times 2^{18}$, were:

- 840 (5 cases)
- $836.0625 = 13377/4^2$ (1 case)
- 836 (1 case)
- 833 (2 cases, known)

Our decomposition program found that the first 7 matrices were indecomposable, but the last two decomposed (as expected). The running time was only 15 seconds (on one processor of orac).

The 9 matrices all had different characteristic equations, so we only had to consider the weakly self-dual case ($G = H$).

One of the solutions

Our program found the following factor of G_1:

$$R = \begin{pmatrix}
- & 1 & 1 & 1 & 1 & 1 & 1 & - & - & - & - & - & - & - & - & - & - & - & - \\
1 & 1 & 1 & 1 & 1 & 1 & - & 1 & 1 & - & 1 & 1 & - & 1 & 1 & - & - & - & - \\
1 & 1 & 1 & 1 & 1 & - & 1 & 1 & - & 1 & 1 & - & 1 & 1 & - & 1 & - & - & - \\
1 & 1 & 1 & 1 & - & 1 & 1 & - & 1 & 1 & - & 1 & 1 & - & 1 & 1 & - & - & - \\
1 & 1 & 1 & - & 1 & 1 & 1 & - & - & 1 & - & 1 & - & 1 & - & - & 1 & 1 & 1 \\
1 & 1 & - & 1 & 1 & 1 & 1 & 1 & - & - & - & - & 1 & - & 1 & - & 1 & 1 & 1 \\
1 & - & 1 & 1 & 1 & 1 & 1 & - & 1 & - & 1 & - & - & - & - & 1 & 1 & 1 & 1 \\
- & 1 & 1 & - & - & 1 & - & 1 & 1 & 1 & 1 & - & 1 & - & - & - & 1 & 1 & - \\
- & 1 & - & 1 & - & - & 1 & 1 & 1 & 1 & 1 & 1 & - & - & - & - & 1 & - & 1 \\
- & - & 1 & 1 & 1 & - & - & 1 & 1 & 1 & - & 1 & 1 & - & - & - & - & 1 & 1 \\
- & - & - & - & 1 & 1 & 1 & 1 & 1 & - & - & 1 & 1 & 1 & - & 1 & 1 & - & - \\
- & - & - & - & 1 & 1 & 1 & 1 & - & 1 & 1 & 1 & - & - & 1 & 1 & - & 1 & - \\
- & - & - & - & 1 & 1 & 1 & - & 1 & 1 & 1 & - & 1 & 1 & 1 & - & - & - & 1 \\
- & - & 1 & 1 & - & - & 1 & - & - & - & 1 & 1 & 1 & 1 & 1 & - & 1 & 1 & - \\
- & 1 & 1 & - & 1 & - & - & - & - & - & 1 & 1 & 1 & - & 1 & 1 & 1 & - & 1 \\
- & 1 & - & 1 & - & 1 & - & - & - & - & 1 & 1 & 1 & 1 & - & 1 & - & 1 & 1 \\
- & 1 & 1 & - & - & - & 1 & 1 & 1 & - & - & - & - & 1 & 1 & 1 & - & 1 & 1 \\
- & 1 & - & 1 & 1 & - & - & - & 1 & 1 & - & - & - & 1 & 1 & 1 & 1 & 1 & - \\
- & - & 1 & 1 & - & 1 & - & 1 & - & 1 & - & - & - & 1 & 1 & 1 & 1 & - & 1
\end{pmatrix}$$

Exercise: Show that R is equivalent to Smith's R_1.

Summary for order 19

The maximal determinant is $B = 833 \times 4^6 \times 2^{18}$ with three (previously known) inequivalent solutions and two (previously known) Gram matrices.

There may be more (inequivalent) solutions, but there are no other Gram matrices of determinant B^2.

Hence all solutions must be weakly self-dual.

The maxdet problem is now solved for $n \leq 21$; the smallest unknown case is $n = 22$.

Order 37

37 is much larger than 19. However, there is hope because $37 = 1 \bmod 4$ and the maxdet matrix of order 37 must have determinant close to the Barba bound.

Maxdet matrices are not known for orders 29 or 33, but they are known for orders 25 and 41, since in these cases the Barba bound can be attained — by Brouwer's construction for order 25, and from the symmetric block design with parameters (41, 16, 6) for order 41.

37 in some sense is "close" to 41.

Results for order 37

Here we took $B = 72 \times 9^{17} \times 2^{36}$. There is a known $\{\pm 1\}$ matrix R with determinant B (93.6% of the Barba bound), found by OS (2003). The corresponding Gram matrix is

$$G = \begin{pmatrix}
37 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 1 & \dots & 1 \\
5 & 37 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \dots & 1 \\
5 & 1 & 37 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \dots & 1 \\
5 & 1 & 1 & 37 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \dots & 1 \\
5 & 1 & 1 & 1 & 37 & 1 & 1 & 1 & 1 & 1 & 1 & \dots & 1 \\
5 & 1 & 1 & 1 & 1 & 37 & 1 & 1 & 1 & 1 & 1 & \dots & 1 \\
5 & 1 & 1 & 1 & 1 & 1 & 37 & 1 & 1 & 1 & 1 & \dots & 1 \\
5 & 1 & 1 & 1 & 1 & 1 & 1 & 37 & 1 & 1 & 1 & \dots & 1 \\
5 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 37 & 1 & 1 & \dots & 1 \\
5 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 37 & 1 & \dots & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 37 & \dots & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \dots & 37
\end{pmatrix}$$

Details for order 37

PZ's Gram-finding program took 77 hours (on one orac processor) to find 807 putative Gram matrices (including the known one on the previous slide). There were 489 different characteristic polynomials, and 1528 pairs (G, H) to consider.

Our decomposition program took 52 minutes to show that 806 of the putative Gram matrices did not decompose (in no case could more than two rows of R be constructed).

It failed to terminate on the remaining case after running for 147 hours and exploring $> 1.7 \times 10^8$ nodes, reaching level 26, but (fortunately) we already know that this matrix does decompose.

A randomised search also failed (after 151 hours). It estimated a tree size (to level 26) greater than 3×10^{14}.

Summary for order 37

The maximal determinant is $B = 72 \times 9^{17} \times 2^{36}$.

There is a unique (up to equivalence) Gram matrix G with determinant B^2.

There may be other (inequivalent) solutions — the known one was constructed by OS from a "doubly 3-normalized" Hadamard matrix of order 36.

Brief summary of other results, $n = 1 \bmod 4$

- We have (easily) confirmed the known maximal determinants for $n = 5, 9, 13, 17, 21, 25$.
- For $n = 29$, with lower bound 320 [omitting the factor $7^{12} \times 2^{28}$], upper bound 370, we improved the upper bound to < 330, disproving the conjectured 336. (Generated 5962 putative Gram matrices in 542 hours and checked them in 19 hours.)
- For $n = 33$, with lower bound 441 [omitting the factor $8^{14} \times 2^{32}$], upper bound 516, we improved the upper bound to < 471. (Checked 9054 putative Gram matrices in 210 hours.)
- For $n = 45$, with lower bound 83 [omitting the factor $11^{21} \times 2^{44}$], upper bound 104, we improved the bounds to $[89, 99)$. (New lower bound due to WO, upper bound by checking 1495 putative Gram matrices.)

Brief summary of other results, $n = 3 \bmod 4$

- We have (easily) confirmed the known maximal determinants for $n = 7, 11, 15$.
- For $n = 23$, we have generated some putative Gram matrices with determinants larger than $(42411 \times 5^6 \times 2^{22})^2$, but none decompose. The Gram-finding phase is incomplete.
- For $n = 23$ and $n = 27$ our program can decompose the Gram matrix corresponding to the best known (possibly but not proved optimal) R.

Future work

A known weakness of our decomposition program is that it does not use the full automorphism groups of G and H when pruning. There is certainly scope for improvement here.

It would also be useful to write a parallel version of our decomposition program, so several processors can share the work of searching the tree in difficult cases. This has been done for the Gram-finding program, but in a fairly primitive way which could be improved to balance the load better. We expect similar load-balancing problems in a parallel version of the decomposition program.

Hasse-Minkowski

Using the Hasse-Minkowski theorem, it can often be shown that a putative gram matrix G does not decompose into $RR^T = R^T R$ even if R is allowed to be a matrix over the *rationals*. Our program does not currently take advantage of this.

References

G. Barba, Intorno al teorema di Hadamard sui determinanti a valore massimo, *Giorn. Mat. Battaglini* **71** (1933), 70–86.

A. E. Brouwer, *An infinite series of symmetric designs*, Math. Centrum, Amsterdam, Report ZW 202/83 (1983).

T. Chadjipantelis, S. Kounias and C. Moyssiadis, The maximum determinant of $21 \times 21(+1,-1)$-matrices and D-optimal designs, *J. Statist. Plann. Inference* **16** (1987), 167–178.

J. H. E. Cohn, Almost D-optimal designs, *Utilitas Math.* **57** (2000), 121–128.

H. Ehlich, Determinantenabschätzungen für binäre Matrizen mit $N \equiv 3 \bmod 4$, *Math. Z.* **84** (1964), 438–447.

William P. Orrick, The maximal $\{-1,1\}$-determinant of order 15, *Metrika* **62**, 2 (2005), 195–219.
`http://arxiv.org/abs/math/0401179`

References continued

William P. Orrick, On the enumeration of some D-optimal designs, *J. Statist. Plann. Inference* **138** (2008) 286–293. http://arxiv.org/abs/math/0511141v2

William P. Orrick, *The Hadamard maximal determinant problem*, http://www.indiana.edu/~maxdet/

William P. Orrick and Bruce Solomon, Large determinant sign matrices of order $4k + 1$, *Discrete Math.* **307** (2007), 226–236. http://arxiv.org/abs/math/0311292v1

Judy-anne H. Osborn, *The Hadamard Maximal Determinant Problem*, Honours Thesis, University of Melbourne, 2002, 142 pp. http://wwwmaths.anu.edu.au/~osborn/publications/pubsall.html

Warren D. Smith, *Studies in Computational Geometry Motivated by Mesh Generation*, Ph. D. thesis, Princeton University, 1988.