Algorithmic Aspects of Integral Designs

Alan Hartman

Discrete Optimization Consulting Pty. Ltd.
1 Irene Place,
Prahran, VIC. 3181
Australia

IBM Israel Science and Technology
MATAM
Haifa, 31905
Israel
email: alan_hartman@vnet.ibm.com

June 1997

Abstract

A *t-design* with parameters $t-(v,k,\lambda)$ is a set X of cardinality v, together with a collection of subsets of X of size k called blocks. The number of blocks containing any particular t-subset of X is a constant λ. The simplest t-designs are regular multi-graphs ($t=1$, $k=2$, v= number of vertices, λ = degree of regularity). Necessary conditions on the four parameters for the existence of t-designs are simply obtained by counting the number of blocks containing a particular i-subset of X for any $0 \le i \le t$. In general these necessary conditions are not sufficient for the existence of t-designs. However these simple counting conditions are sufficient for the existence of so-called "integral designs" which are an extension of the definition of t-designs, allowing blocks to have negative as well as non-negative multiplicities. The sufficiency of the elementary necessary conditions for the existence of integral designs was proved independently by Graver and Jurkat, and Wilson in the early 1970s.
The proof by Wilson is algorithmic in nature. We discuss implementations of his construction algorithm which aim to both reduce the number of blocks with negative multiplicities, and also reduce the absolute value of the most negative multiplicity - thus hoping to produce integral designs which are "close" to designs.

The author wishes to thank Professor Lou Caccetta and the School of Mathematics and Statistics at Curtin University of Technology for their hospitality and support during the preparation of this paper.

Introduction

We define a *t*-design with parameters $t-(v,k,\lambda)$ to be a set X of cardinality v, called the set of points, together with a collection, B, of subsets of X of size k called blocks. The number of blocks containing any particular t-subset of X is a constant λ. We use the word collection for B advisedly, since we wish to allow any block, b, to occur a positive number, $m(b)$, times in the design. The integer $m(b)$ is called the multiplicity of b. We also note that if $\lambda > 0$, then we must have $0 \le t \le k \le v$, for the definition to make sense.

Let us denote the set of all k-subsets of X by $\binom{X}{k}$. Our formal definition of a $t-(v,k,\lambda)$ design is then a vector of non-negative integers $\mathbf{m} = (m(b): b \in \binom{X}{k})$ with the property that $\sum_{b \supset T} m(b) = \lambda$ for all $T \in \binom{X}{t}$.

If we count the number, n_j, of blocks containing a particular subset of J of cardinality j for any $0 \le j \le t$, then we observe that

$$n_j = \lambda \binom{v-j}{t-j} / \binom{k-j}{t-j}$$

since the number of t-subsets containing J is $\binom{v-j}{t-j}$, each of which occurs λ times, but each block containing J is counted $\binom{k-j}{t-j}$ times. Thus we have that a necessary condition on the integers t, v, k, and λ for the existence of a $t-(v,k,\lambda)$ design is that

(1) $\lambda \binom{v-j}{t-j} \equiv 0 \left(\mathrm{mod} \binom{k-j}{t-j} \right)$ for all $0 \le j \le t$.

Graver and Jurkat [1] and Wilson [2] proved that there exists a constant $N(t,v,k)$ such that for all λ larger than $N(t,v,k)$ the necessary conditions (1) are sufficient for the

existence of a $t-(v,k,\lambda)$ design. In doing so they proved a more general theorem on the existence of so-called integral designs. Let X be a set of cardinality v. An *integral design* with positive integer parameters t, v, k, and target integer vector $\lambda = (\lambda(T): T \in \binom{X}{t})$, is an integer vector $\mathbf{m} = (m(b): b \in \binom{X}{k})$ with the property that $\sum_{b \supset T} m(b) = \lambda(T)$ for all $T \in \binom{X}{t}$.

Clearly a design is just an integral design with a constant positive target vector, and where each of the block multiplicities, $m(b)$, is non-negative.

Repeating the argument used above, we see that necessary conditions on the parameters and target vector for the existence of an integral design are that:

(2) $\sum_{T \supset J} \lambda(T) \equiv 0 (\mathrm{mod} \binom{k-j}{t-j})$ for all subsets $J \in \binom{X}{j}$, and for all $0 \leq j \leq t$,

since the left hand side over-counts the sum of the multiplicities of the blocks containing J by a factor of $\binom{k-j}{t-j}$.

It will also be convenient for us to consider an integral design as an integer solution to a matrix equation. Let us define A(t,k,v) to be the 0-1 matrix with $\binom{v}{t}$ rows indexed by the t-subsets of X, and $\binom{v}{k}$ columns indexed by the k-subsets of X. The entry in the (T,K)-th position will be 1 if $T \subseteq K$, and 0 otherwise. Now the definition of an integral design is just an integer vector $\mathbf{m}$, which is a solution to the matrix equation

(3) $A(t,k,v)\mathbf{m} = \lambda$.

Graver and Jurkat's, and Wilson's result is then simply stated as: Equation (3) has an integer solution if and only if the necessary conditions (2) hold. Their proof of this

theorem is by induction on v and t. Like all induction proofs, one can convert this proof into a recursive function which constructs integral designs. This paper describes an implementation of this recursive function, and the results obtained by exploiting the freedom inherent in the implementation details of this algorithm.

The Induction Proof

The proof begins by noting that if $v \le k+t$, then the rows of A(t,k,v) are linearly independent, and when $v = k+t$, then the matrix is invertible. We omit the details of the proof that the solution to equation (3) is integral, but we note that it is a relatively simple exercise to invert the matrix A and compute the resulting design **m**, in the case when $v = k+t$. We also note that the solution is unique.

When $t = 0$ then the target vector is just an integer λ, and the matrix A(0,k,v) is all ones, so a solution to equation (3) in this case is just any integer vector of length $\binom{v}{k}$ whose sum is λ. In this case there are many possible solutions.

In the case where $v > k+t$, and $t > 0$, we proceed by induction. Choose a point $x \in X$, and define a new target vector (called the *derived* target) $\lambda_d(T - \{x\}) = (\lambda(T))$ on only those t-subsets of X which contain x. The new target vector is of length $\binom{v-1}{t-1}$. It is not difficult to check that λ_d satisfies the necessary conditions with parameters $t-1$, $k-1$, and $v-1$. By induction we can solve equation (3) to obtain a *derived* design $\mathbf{m_d}$ of length

$\binom{v-1}{k-1}$. We now extend $\mathbf{m_d}$ to a vector $\mathbf{M_d}$ of length $\binom{v}{k}$ by setting

$M_d(b) = m_d(b - \{x\})$ for each block b such that $x \in b$, and setting $M_d(b) = 0$ otherwise.

The next step is to form a second new target vector (called the *residual* target) λ_r defined by $\lambda_r = \lambda - A(t,k,v)\mathbf{M_d}$. By construction, the residual target vector will be 0 on all t-subsets which contain x. Let us now shorten the residual target by deleting those zeroes corresponding to t-subsets containing x. This shortened residual target now can be shown to satisfy the necessary conditions with parameters t, k, and $v-1$. Again, by induction, there exists a design vector $\mathbf{m_r}$ of length $\binom{v-1}{t}$ which can be extended to a vector $\mathbf{M_r}$ of length $\binom{v}{t}$ by setting $M_r(b) = m_r(b)$ for each block b such that $x \notin b$, and setting $M_r(b) = 0$ otherwise.

We now have constructed a solution $\mathbf{m} = \mathbf{M_d} + \mathbf{M_r}$ to the original equation, since

$A(t,k,v)\mathbf{m} = A(t,k,v)\mathbf{M_d} + A(t,k,v)\mathbf{M_r} = \lambda - \lambda_r + \lambda_r = \lambda$.

The Algorithm

We now give an algorithmic version of this induction proof using a recursive function **Design(t,k,v,λ)** which receives the integers $t \geq 0$, $k \geq t$, and $v \geq k + t$, and an integer vector **λ**, of length $\binom{v}{t}$, satisfying the necessary conditions (2). The output of the

function is an integer vector **m** of length $\binom{v}{k}$ which is a solution to equation (3). To remove ambiguities we will consider X to be the set $\{0,1,...,v\text{-}1\}$, and assume that the subsets which index the vectors and matrices are ordered lexicographically.

The pseudo code for this function is:

```
if t = 0 then return(Integer_Vector_Of_Sum(λ, v, k, ...)) else

if v = k + t then return(A(t, k, v)^-1 λ) else

begin
   x = Choosepoint(v, ...)
   λd=Construct_Derived_Vector(x, t, v, λ)
   Md= Extend1(x, Design(t-1, k-1, v-1, λd))
   λr = Contract(x, λ – A(t, k, v)Md)
   Mr = Extend2(x, Design(t, k, v-1, λr))
   return(Md + Mr)
end
```

The functions **Construct_Derived_Vector, Extend1, Contract,** and **Extend2,** are fully described in the previous section and have obvious implementations. The functions **Integer_Vector_Of_Sum,** and **Choosepoint** are capable of many implementations. The main focus of this research was on different implementations of these two functions, with the aim of minimizing both the number of negative multiplicities in the design, and the absolute value of the most negative multiplicity.

The function **Choosepoint** outputs a point in the set X of cardinality v, so a minimal input requirement is the integer v. We experimented with five implementations of

Choosepoint:

Choosepoint0(v) which always returns the first member of the set,

ChoosepointV(v) which always returns the last member of the set,

ChoosepointR(v) which returns a random member of the set,

ChoosepointMin(v,λ) which returns the first point x which minimizes $\sum_{x \in T} \lambda(T)$,

ChoosepointMax(v,λ) which returns the first point x that maximizes $\sum_{x \in T} \lambda(T)$.

We also experimented with five implementations of **Integer_Vector_Of_Sum** (below shortened to **IVOS**). This function outputs an integer vector of length $\binom{v}{k}$ whose sum is the integer λ. The first three implementations are straightforward:

IVOS_All_Together(λ,v,k) outputs the vector [λ,0,0,...,0].

IVOS_Lex_Order(λ,v,k) outputs the vector with 1s or -1s (depending on the sign of λ) in the first |λ| positions, and 0 elsewhere.

IVOS_Random(λ,v,k) starts with the zero vector, chooses |λ| positions in the vector at random, and increments each one by 1 or -1, depending on the sign of λ.

The initial runs with just these implementations yielded the depressing results that the best pair of implementations involved choosing the λ-minimal point to delete (**ChoosepointMin**), then choosing blocks at random (**IVOS_Random**). This is depressing since the choice of the λ-minimal point is the choice which does the least damage, and one would hope that human ingenuity was superior to simply choosing blocks at random. To overcome our angst, we determined to find implementations of **Integer_Vector_Of_Sum** which utilised the information lost in the recursive

implementation of the **Design** function.

We introduced a new vector parameter to the design function entitled **BS** (for block score). The vector **BS** is of length $\binom{v}{k}$ and is initialized by $\mathbf{BS}(b) = \sum_{T \subseteq b} \lambda(T)$. When making the call to **Design** to construct the derived design, we form the derived block score vector by shortening it, just as the target vector is shortened. The residual block score vector is constructed so as to discourage the use of blocks which have a large intersection (of size $k-1$) with blocks in the derived design. This is achieved by setting $\mathbf{BS}_r(b) = \mathbf{BS}(b) - \sum_{x \in b} m_d(b - \{x\})$.

This is a crude method of scoring the blocks, but the improvement in the results achieved was significant. The function **IVOS_First_High_Score** outputs a vector with $|\lambda|$ 1s or -1s in the first positions with the highest block score, and the function **IVOS_Random_High_Score** chooses a set of $|\lambda|$ positions of highest score at random to increment by 1 or -1 depending on the sign of λ.

Computational Results

The program was run with all twenty-five combinations of the five **Choosepoint** and **Integer_Vector_Of_Sum** implementations. Those implementations with a random component were run three times each and the best result was recorded. The results of a run were summarized in two integers, the absolute value of the most negative block multiplicity in the solution, and the number of blocks with a negative multiplicity. Thus a score of (0,0) is best possible. We summarize the results obtained for thirteen sets of design parameters in the following table.

t-(v,k,λ)	(most negative block multiplicity, number of negative blocks)
2-(6,3,2)	15 Algorithm combinations scored (0,0)
2-(7,3,1)	14 Algorithm combinations scored (0,0)
2-(7,3,2)	11 Algorithm combinations scored (0,0)
2-(9,3,1)	5 Algorithm combinations scored (0,0)
2-(13,3,1)	3 Algorithm combinations scored (0,0): CR-IFH, CV-IRH, and CLm-IRH
2-(10,4,2)	2 Algorithm combinations scored (1,7): CV-IRH, and CLm-IRH 2 Algorithm combinations scored (2,6): CLm-IFH, and CR-IRH
2-(15,3,1)	4 Algorithm combinations scored (0,0): C0-IFH, CV-IFH, CLM-IFH, and CLm-IRH
2-(10,3,2)	5 Algorithm combinations scored (0,0)
2-(12,4,3)	CLM-IRH scored (1,15), CR-IRH scored (2,12), and CLm-IRH scored (3,8)
2-(13,4,1)	CLm-IRH scored (0,0)
3-(8,4,1)	13 Algorithm combinations scored (0,0)
3-(10,4,1)	3 Algorithm combinations scored (0,0): CR-IRH, C0-IRH, and CR-IFH
3-(14,4,1)	CLm-IRH scored (1,25)

The codes used to describe the algorithms are as follows:

Code	Algorithm	Code	Algorithm
C0	**Choosepoint0**	IL	**IVOS_Lex_Order**
CV	**ChoosepointV**	IA	**IVOS_All_Together**
CR	**ChoosepointR**	IR	**IVOS_Random**
CLm	**ChoosepointMin**	IFH	**IVOS_First_High_Score**
CLM	**ChoosepointMax**	IRH	**IVOS_Random_High_Score**

Conclusions and Future Directions

It is apparent that the best combination of algorithms is currently **ChoosepointMin** and **IVOS_Random_High_Score.** This is an improvement on the situation before a block score vector was introduced to the **Design** routine. There is definitely room to improve the scoring of the blocks, refining the computation of scores to a point where the best performing algorithms will be **ChoosepointMax** and either of the **High_Score** algorithms. There is probably also a case for experimenting with mixed strategies using one algorithm in some circumstances, and another at a later stage in the building of the design.

The application of this technique for the construction of t-designs is still in its infancy, but I believe that it has great potential for constructing small designs of the many types needed for the recursive constructions used in design theory. It is readily applicable to the construction of trades, group divisible designs, holey designs, and any other design whose definition can be expressed in terms of a target vector of t-subsets

References

[1] Graver J.E. and Jurkat W.B., **The module structure of integral designs**, Journal of Combinatorial Theory 15A(1973), 75-90.

[2] Wilson R.M., **The necessary conditions for t-designs are sufficient for something,** Utilitas Mathematica 4(1973), 207-215.

(Received 1/9/97)