

**APPLICATIONS OF COMBINATORIAL DESIGNS
IN COMPUTER SCIENCE**

By

Charles J. Colbourn

and

Paul C. van Oorschot

IMA Preprint Series # 400

March 1988

Applications of Combinatorial Designs in Computer Science

Charles J. Colbourn and Paul C. van Oorschot

Department of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
CANADA

ABSTRACT

The theory of combinatorial designs has been used in widely different areas of computation, in the design and analysis of both algorithms and hardware. Combinatorial designs capture a subtle balancing property which is inherent in many difficult problems, and hence can provide a sophisticated tool for addressing these problems. The role of combinatorial designs in solving many problems which are basic to the field of computing is explored in this paper. Case studies of many applications of designs to computation are given; these constitute a first survey which provides a representative sample of uses of designs. More importantly, they suggest paradigms in which designs can be used profitably in algorithm design and analysis.

1. The Background

The interaction between mathematics and computer science has proved fundamental to the vitality of both fields over the last thirty years; nowhere is this more true than in the area of combinatorics and graph theory. Graph-theoretic tools arise in virtually every area of computational study: network design and analysis, database theory, artificial intelligence, complexity theory and matrix computations are just a few of the areas where sophisticated graph-theoretical tools are routinely used. By the same token, computational studies have focussed much research attention on graph theory; the area of combinatorial optimization provides a host of examples of combinatorial theorems which are both elegant mathematically and practical computationally. In one sense, the fruitfulness of this interplay is to be expected, since graphs are natural models of the finite structures which computers by their very nature manipulate.

The role of combinatorics in computer science, however, is not limited to the role of graph theory. Many problems involve finite structures which are highly constrained; the mathematics of such structures then comes into play. One very active area of

combinatorics which deals primarily with systems of sets satisfying some "balance" constraints is combinatorial design theory. The study of combinatorial designs dates back over a century and a half. In the 1930's, combinatorial design theory was driven largely by applications in experimental design theory; this close connection remains today (see, for example, Raghavarao (1971)). In the 1950's, a multitude of other applications arose in the theory of error-correcting codes; designs remain fundamental in coding theory (see, for example, MacWilliams and Sloane (1978)). These connections are for the most part well understood and widely used.

At present, the role of combinatorial designs in computer science is less well appreciated. However, this role is both more substantial and more varied than one might first expect. Our thesis is that the tools of combinatorial design theory form an integral part of the equipment needed to solve problems of computation effectively. In fact, we shall see that computer science and combinatorial design theory have many areas of intersection, and that each profits by using results from the other. The large role which algorithms play in combinatorial design theory is well documented in a recent survey by M.J. Colbourn (1985). However, the role of combinatorial designs in computation has typically been underestimated, at least in part due to the lack of any survey on this subject.

Our aim is twofold. A primary goal is to survey applications which have appeared in the computing literature, with emphasis on the "bread-and-butter" problems of computation: sorting, searching, selection, and the like. More importantly, we use case studies to suggest paradigms for problems in which designs prove useful. Our intent here is not to solve new problems using designs, nor is it to completely characterize problems where designs are useful. Rather, it is to provide a framework for understanding the use of designs as a tool in algorithm design and analysis.

In the remainder of the paper, we proceed as follows. First, a basic introduction to combinatorial design theory is presented. Then we explore problems of computation in which designs have been profitably employed in the past; while each case is tied to a particular application, each illustrates a class of applications in which similar design-theoretic techniques are useful. We explore applications to the design of access switches (section 3), threshold schemes (section 4), authentication codes (section 5), filing schemes (section 6), parallel sorting (section 7), design and analysis of algorithms (section 8), lower bounds for complexity (section 9), interconnection networks (section 10), distributed algorithms (section 11), and relational databases (section 12). Finally, we conclude with a discussion of paradigms which arise from the case studies, and thereby suggest further applications for designs.

2. Combinatorial Design Theory: An Introduction

In this section, we provide enough background in combinatorial design theory to enable the reader to appreciate the role of combinatorial designs in the problems which we explore in subsequent sections. A complete understanding of the applications requires a somewhat more detailed introduction; we defer this to the appendix, to which the interested reader can refer as needed. Much fuller treatments of combinatorial design theory appear in recent texts by Beth, Jungnickel and Lenz (1985), by Hughes and Piper (1985), and by Street and Street (1987).

Let V be a finite set of v elements. A *set system* \mathcal{B} on V is a collection of subsets of V ; the set system is *simple* when \mathcal{B} does not contain any subset more than once. Set systems are widely studied under the name *hypergraphs*. We are interested in set systems which exhibit various kinds of "regularity" or "balance"; to be specific, we require some further definitions.

Sets in \mathcal{B} are called *blocks*; the number of blocks is denoted $b = |\mathcal{B}|$. K denotes the set of *blocksizes* $\{|B| : B \in \mathcal{B}\}$. When $K = \{k\}$, the set system is called *k-uniform*. Each element $x \in V$ appears in some subset $\mathcal{B}_x \subseteq \mathcal{B}$. The *replication number* $r_x > 0$ is $|\mathcal{B}_x|$, and the set of replication numbers is $R = \{r_x : x \in V\}$. Every subset $S \subseteq V$ appears as a (not necessarily proper) subset of a number of the sets in \mathcal{B} ; the number of sets in \mathcal{B} containing S is denoted $\lambda(S)$ and is termed the *index* of S in \mathcal{B} . We define the *t-index set* of a set system (V, \mathcal{B}) to be the set $\{\lambda(S) : |S| = t, S \subseteq V\}$. A set system is called *t-balanced* if the *t-index set* contains a unique value λ_t and $\lambda_t > 0$. A 1-balanced set system is simply one in which there is a single replication number r , and hence $r = \lambda_1$. Every set system is 0-balanced, with $b = \lambda_0$. Observe that any *t-balanced k-uniform set system* is also $(t-1)$ -balanced.

Let us consider a small set system. Let $V = \{0,1,2,3\}$ and $\mathcal{B} = \{\{0,1,2\}, \{0,1,3\}, \{0,2,3\}, \{1,2,3\}\}$. This set system has $v=4$, $b=4$, $K=\{3\}$ and $R=\{3\}$. It is, in fact, 2-balanced (with 2-index 2) and also 3-balanced (with 3-index 1). This small set system illustrates that it is possible to have a unique block size, a unique replication number, and a unique *t-index*. Such set systems provide the most basic type of combinatorial designs.

A (*balanced incomplete*) *block design* is a pair (V, \mathcal{B}) where \mathcal{B} is a *k-uniform set system* on V which is 2-balanced with index λ . (V, \mathcal{B}) is typically termed a (v, k, λ) -design; it is an easy exercise to see that the (unique) replication number r is determined, as is the number b of blocks. Block designs are set systems in which the appearance of unordered pairs is uniform; the natural extension is to set systems which are *t-balanced*. A *t-design* (V, \mathcal{B}) , or more precisely, a $t-(v, k, \lambda)$ design, is a *k-uniform set system* with $|V| = v$ which is *t-balanced* with *t-index set* $\{\lambda\}$. *Trivial t-designs* arise by taking $v=k$ or $k=t$; hence our small example is a trivial 3-design (but it is a non-trivial 2-design or block design).

These very regular set systems are the main topic of study in combinatorial design theory. Main topics of research in design theory which are used in the applications to follow include necessary and sufficient conditions for existence, symmetric designs, duals of designs, generalizations of designs (packings, coverings, and pairwise balanced designs), resolutions of designs and automorphisms of designs. Each of these topics is discussed in a subsection of the appendix. We summarize a few of the most basic points here.

The basic existence question is to decide for which parameters t, v, k, λ there exists a t -(v, k, λ) design. Numerical necessary conditions rule out many parameter sets. For those remaining, existence is established for $t=2$ and k, λ fixed as $v \rightarrow \infty$; other than this, many sporadic examples and infinite families are known, but existence remains far from settled in general.

Designs must have at least as many blocks as elements, and hence the designs with the same number of blocks as elements are widely used: these designs are *symmetric*. Special cases which we use are for $\lambda=1$ (projective planes), $\lambda=(q^m-1)/(q-1)$ for q a prime power (projective geometries), and $\lambda=(v-3)/4$ (Hadamard designs).

Designs are closely related to other combinatorial objects which we also encounter. Packings and coverings relax the balance requirement on subsets, so that each t -subset appears at most (for packings) or at least (for coverings) λ times. Pairwise balanced designs relax the requirement that the block size is uniform, and allow a number of different block sizes. Set packings balance the sizes of intersections of blocks, by requiring that blocks not intersect in more than λ elements; exact set packings require intersection in precisely λ elements. These variants all arise in part because existence of t -designs remains unsettled.

A vast number of properties of designs have been studied; we employ two in a substantial way. The first property of interest is that for some designs, one can partition the blocks into *parallel classes*, which are sets of disjoint blocks whose union is the set of elements in the design. A design with such a partitioning is *resolvable*. We also employ designs with non-trivial automorphisms (symmetries).

This thumbnail description of design theory is sufficient to appreciate the main ideas in the applications to follow; readers desiring a better understanding can find the required design-theoretic definitions and background in the appendix.

3. Magnetic Core Access Switches

In this section, we explore the use of block designs and pairwise balanced designs in the design of magnetic core access switches; this application illustrates the use of designs in many selection problems.

Memory design was dominated for many years by magnetic core memories. Minnick and Haynes (1962) provide a comprehensive survey of the design of access switches. We recall the main points here. There are n magnetic *cores* c_1, \dots, c_n , which are primitive bistable memory devices. There are m *wires* which are capable of carrying a current; each wire can be "wound through" a core, either positively or negatively. We let ρ_{ij} be 1, -1 or 0 according to whether w_j winds c_i positively, negatively, or not at all. When each wire w_j carries current σ_j , core c_i receives the signal $\sum_{j=1}^m \sigma_j \rho_{ij}$. Core c_i is *selected* if and only if the signal received exceeds a specified *activation threshold* A . Notice that since a core is wound by many wires in general, there is a *load-sharing* permitted, by which each wire need only deliver a portion of the activation current A ; the *load-sharing factor* is simply A divided by the largest σ_j used, $j \leq m$.

Nonzero signals of strength less than A are termed *noise*. The operation of a memory requires that we be able to select each core separately (in order to read its value or write a value into it). Hence, for each core c_i , we require a set of *wire activations* $\alpha_i = (\alpha_{i1}, \dots, \alpha_{im})$ so that $\sum_{j=1}^m \alpha_{ij} \rho_{kj} \geq A$ if and only if $k = i$. Variations in the physical wires used cause a difficulty here, in that noise may be mistaken for an activation signal. One therefore introduces a *noise threshold* $N < A$ and requires that the signal received at core c_k when c_i is selected does not exceed the threshold N .

The basic goals in this problem are easily stated.

1. The number of wires is to be minimized.
2. The noise threshold N is to be minimized.
3. The load-sharing factor is to be maximized.

The third goal is intended to reduce electrical interference, and allow wires to be placed closer together in the actual circuit.

Naturally, these three goals are conflicting; this leads to a number of important tradeoffs. A number of variants of this basic scenario have been studied; we examine some of them here.

3.1. Positive Winding

In this case, we require that $\rho_{ij} \in \{0, 1\}$ and that $\sigma_j \in \{0, 1\}$. Let $X = \{c_1, \dots, c_n\}$ be the set of cores. Since each core is wound only positively, we let B_j be the set of cores wound positively by wire w_j , and $\mathcal{B} = \{B_1, \dots, B_m\}$. Dually, if $W = \{w_1, \dots, w_m\}$ is the set of wires, we let C_i be the set of wires wound through core c_i , and $\mathcal{C} = \{C_1, \dots, C_n\}$.

Now let us suppose that the activation threshold is A , and the noise threshold is N . (X, \mathcal{B}) is then a partial PBD with replication number at least A , and index at most N . Dually, (W, \mathcal{C}) is a set packing with block size at least A and intersection numbers at most N . One primary goal is to minimize m for fixed n , or equivalently to maximize n for fixed m . Since (W, \mathcal{C}) is a set packing, n is maximized as a function of m when (W, \mathcal{C}) is in fact a symmetric block design.

In the practical setting, notice that a core now receives a signal of at least A if it is selected, but can receive any signal in the range $[0, N]$ when it is not selected. In the extreme case, we may not be willing to tolerate any noise. The only zero-noise switch under the present assumptions requires that each wire wind exactly one core. Naturally then no load-sharing is occurring.

3.2. Positive Winding With Bias

In this case, only one change is made: a *bias wire* w_{m+1} is wound negatively through each core. We allow the bias wire to carry a (possibly) larger current, the *bias signal*; all other wires still carry signals of 0 or 1. With the simple addition of one bias wire, it is possible to construct zero-noise load-sharing switches.

Let (X, \mathcal{B}) be, as before, a partial PBD with minimum replication r and index at most λ ; using a bias signal of λ and an activation threshold of $r - \lambda$ shifts all noise into the range $[-\lambda, 0]$. This scheme is zero-noise if and only if (X, \mathcal{B}) is a PBD. Hence we can meet the minimum noise goal by taking any PBD here. Having decided on a PBD, we now want to minimize the number of wires, $|\mathcal{B}|$, and to maximize the load-sharing factor. This is accomplished by taking (X, \mathcal{B}) to be a symmetric design.

Blachman (1956) describes a similar scheme based on projective planes. His rationale for choosing planes over other symmetric designs is that the number of windings per core remains physically realizable; it is the replication number, plus one for the bias wire. Singleton (1962) later described this scheme in more generality, and explored the fault tolerance that the scheme supports.

Naturally, one may not always be fortunate enough to have the required symmetric design for constructing the switch. Hence two relaxations are of interest. First, we shall retain the zero-noise property, which (as observed above) is done provided that we select (X, \mathcal{B}) to be a PBD, but now allow the load-sharing factor to be reduced. The load-sharing factor is the minimum replication number minus the index. In order to avoid reducing the load-sharing factor too far, suitable PBDs are those having only small variation in block sizes (and hence small variation in replication numbers), while still keeping $|\mathcal{B}|$ acceptably small. PBDs of this sort have recently been studied by Erdős and Larson (1982), who develop solutions by omitting points of a projective plane.

A second relaxation is to tolerate a small amount of noise. For example, if we take (X, \mathcal{B}) to be a partial PBD with constant block size in which each pair appears either $\lambda-1$ or λ times, we can use a bias signal of $(2\lambda-1)/2$ to bound the noise to $1/2$ while retaining good load-sharing properties. Any partial PBD with the two possible indices $\lambda-1$ and λ would serve here, but the best will have largest minimum replication number. The scheme supports generalization to more allowed indices, with a consequent increase in noise. A particular scheme of interest here is obtained by selecting some number p of parallel classes from the affine plane of order n to form (X, \mathcal{B}) . This leads to an activation threshold of $p-1/2$ and a noise threshold of $1/2$. When $p = 2$, this is the widely used "square switch" (Minnick and Haynes (1962)). Singleton (1962) described many variants of this strategy; he considered the dual problem, that of forming set packings with given minimum block size and few allowed intersection numbers.

3.3. Positive and Negative Winding

In this case, $\rho_{ij} \in \{-1, 1\}$ and $\sigma_j \in \{0, 1\}$. We let $B_j = \{c_i : \rho_{ij} = 1\}$ and form (X, \mathcal{B}) as before. Suppose that (X, \mathcal{B}) is a symmetric block design with parameters (v, k, λ) . Selecting all blocks containing c_i delivers a signal of r to c_i , and a signal of $\lambda-(r-\lambda)$ to all other cores. Hence the addition of a bias wire with bias signal $r-2\lambda$ gives an activation threshold of $2r-2\lambda$ and zero noise.

Any symmetric block design leads to such a scheme; to maximize load-sharing, we would opt for a design with $2r-2\lambda$ as large as possible (for fixed v). Hence, in the practical context we would prefer Hadamard designs. Constantine (1958, 1960) first suggested this basic approach, and Marcus (1959) and Chien (1959, 1960) refined it. Singleton (1962) recast much of their work in the context of block designs, as described here.

3.4. Other Switches

Once we depart from the requirement that all wires except the bias carry a unit signal, there is a multitude of possible switches. Many of these are also based on block designs. In fact, Singleton (1962) develops a scheme based on designs which is zero-noise without requiring a bias wire. He also explores the use of set packings in the event that more wires are allowed.

The practical consequence of this work is limited to the design of small memories, primarily because the zero-noise schemes require each core to be wound with a large number of wires. Moreover, there is a difficult geometric problem of realizing the windings in circuitry, which imposes strong constraints on the number (and pattern) of windings. The primary practical merit of the design-theoretic approach is in the description of low-noise switches.

4. Threshold Schemes

In this section, we explore a different use of block designs in solving a selection problem. In transaction-based systems, it is often desirable to have the capability to control the execution of certain important operations (e.g. authorization of cheques within a corporation) or restrict access to secret information (e.g. cryptographic keys). In particular, suppose one wishes to divide an *access privilege* among w people, such that by acting in unison, any t of these w can gain access, but any $t-1$ or fewer cannot. Such a system is called a *threshold scheme*. Threshold schemes were first discussed by Blakley (1979) and Shamir (1979).

Formally, let X be a set of v *shadows* (corresponding to partial privileges), K be a set of m *keys* (to which access is desired), and \mathcal{B} be a set of b distinct w -subsets of X . A (t, w, v) -*threshold scheme* is then a pair (\mathcal{B}, K) together with a mapping $f: \mathcal{B} \rightarrow K$ such that for every t -subset T of X , for all blocks $B \in \mathcal{B}$ such that $T \subseteq B$, $f(B) = K$ for some fixed $K \in K$ (i.e. T distinguishes a unique key), but for $s \leq t-1$, no s -subset S of X determines a unique key (i.e. for no $S \subseteq X$ is $f(B)$ unique for all $B \in \mathcal{B}$ containing S). To "share" a secret key K among w people, a block $B \in \mathcal{B}$ is selected such that $f(B) = K$, and the shadows in B are distributed among the w people. Then any group of t of these w people can uniquely determine K via f .

As noted by Beutelspacher (1987), $t-(v, k, 1)$ designs (i.e. Steiner systems) yield (t, k, v) -threshold schemes directly, as each t -subset of their point set distinguishes exactly one block of the design. Symmetric designs with index λ also yield threshold schemes directly; since any two blocks intersect in exactly λ points, any $\lambda+1$ points determine a unique block (or no block), and hence yield threshold schemes with $t = \lambda+1$.

In a t -threshold scheme, although no group of $s \leq t-1$ people can determine a unique key, it is possible that through collaboration, such a group is able to rule out certain keys from the key set. With this in mind, a (t, w, v) -threshold scheme is said to be *perfect* if no set of $s \leq t-1$ shadows from a block gives any partial information as to which key that block determines. More formally, the probability that an s -subset S of X distinguishes any key $K \in K$ must equal the *a priori* probability that key K is distinguished, i.e. the requirement is $\text{prob}(K | S) = \text{prob}(K)$. Stinson and Vanstone (1988) determine an upper bound on the number m of keys in a perfect (t, w, v) -threshold scheme to be

$$m \leq \frac{v - (t-1)}{w - (t-1)} = M,$$

and prove that $m = M$ if and only if there exists a $t-(v, w, 1)$ design which is partitionable into $(t-1)-(v, w, 1)$ designs.

Stinson and Vanstone go on to actually construct several classes of such *optimal* threshold schemes by making use of several known classes of partitionable Steiner systems. For p a prime congruent to 7 modulo 8, optimal $(3, 3, p+2)$ -threshold schemes

can be constructed by employing a partition due to Schreiber (1973) and Wilson (1974) of a $3-(p+2,3,1)$ design into p $2-(p+2,3,1)$ designs. A second infinite class of optimal threshold schemes with parameters $(3,4,2^{2m})$ can be constructed using a partition due to Zaitsev et al. (1973) of a $3-(2^{2m},4,1)$ design into $2^{2m-1}-1$ $2-(2^{2m},4,1)$ designs. These schemes allow more keys than do previously known threshold schemes for $t = 3$ and $w = 3,4$.

As with the design of core access switches, the balanced appearance of subsets is exploited here to ensure that no partial information ("noise") results.

5. Authentication Codes

Suppose two parties wish to communicate remotely but there is some fear that an "enemy" may attempt to either alter a message in transit or insert a message of his own, passing it off to one party as a message from the other. Authentication codes can be used to minimize the chance of such deception going undetected.

Let the information to be communicated be represented as one of k states from a state set S . The legitimate parties in the communication use some set V of $v \geq k$ messages to represent these states, along with a set E of encoding rules. Each encoding rule provides a one-to-one mapping of the elements from some subset of S to those of some subset of V . The set of states, messages and encoding rules define an *authentication code*. The parties agree (ahead of time) to use some encoding rule $e \in E$; then to communicate some state $s \in S$ contained in the domain of e , the message $e(s)$ is sent. It is assumed that the enemy is unaware which particular rule e is selected, but knows E .

Simmons (1984) notes two probabilities of interest: the probability p_i that an enemy is able to insert a message without the receiver detecting the illegitimate source, and the probability p_a that an enemy is able to alter a legitimate message without detection. Since the legitimate parties agree beforehand on a particular encoding rule, and not all messages are valid under a given rule, an enemy's insertion (alteration) will be detected if the message he inserts (substitutes) is invalid under the encoding rule in use.

Authentication codes which minimize p_i and p_a are desirable. It can be proven that in an authentication code with k states and v messages, $p_i \geq k/v$ and $p_a \geq k-1/v-1$; see Stinson (1988). In fact, a (v,k,λ) design with b blocks and replication value r can be used to construct an authentication code which provides k states and v messages using b encoding rules, and meets these bounds with equality, under the assumption that the states are equally probable and each rule is selected with equal probability. The points of the design correspond to messages, and the blocks specify encoding rules, with each element of a given block determining a unique state. p_i then corresponds to the probability that an inserted message appears as a point in the block corresponding to the selected encoding rule; this probability is therefore r/b , which by

the standard relations governing design parameters is equal to k/v . p_a corresponds to the probability that the original message m and that substituted for it by an enemy both occur in the block corresponding to the selected encoding rule, a block known to contain the point m ; this probability is thus $\lambda/r = k-1/v-1$. This simple construction suggests a natural connection between designs and authentication codes.

An authentication code may in addition provide some degree of secrecy — protection against an enemy extracting state information from an observed message. Authentication without secrecy may be acceptable — for example, for cheque authorization. An authentication code in which a message uniquely identifies a state provides *zero secrecy*; such a code may actually be required, for example, in certain diplomatic situations. In an authentication code providing *perfect secrecy*, sight of a message provides no information about the state it encodes.

Using (v, k, λ) designs, Stinson shows how to construct authentication codes which provide perfect secrecy and meet the bounds on p_i and p_a with equality. These codes provide k states using v messages and bk encoding rules, where b is the number of blocks in the design. Actually, he gives a more general result, using a particular class of set systems known as group divisible designs, which are a generalization of block designs.

Stinson also generalizes an earlier result of Brickell (1984), which employs a class of group divisible designs known as transversal designs to construct zero secrecy authentication codes which provide k states and use kn messages. These codes have $p_i = 1/n = p_a$, hence meeting the bound on p_i with equality and very nearly attaining the bound on p_a .

In addition to minimizing p_i and p_a and achieving the desired secrecy properties, it is desirable to find authentication codes which use a minimal number of encoding rules. Stinson (1987) considers authentication codes which meet the bounds on p_i and p_a with equality, use a minimal number of encoding rules, and provide perfect secrecy in the situation in which an enemy observes *two* messages encoded under same encoding rule. Employing a type of design called a perpendicular array, he establishes the existence of such codes and describes their construction and implementation.

6. File Organization

A *file* is a collection of *records*; each record has a number of *attributes*, and we retrieve records by specifying their attributes. A primary requirement for any file organization is the support of *partial match queries*; here, values for some of the attributes are given and the remainder are left unspecified, and all records matching the values in the specified attributes are to be retrieved. Normally, records are relatively space-consuming objects; hence they are stored on a slower secondary storage device and an *accession number* records their address on this device. Our task, given a partial match query, is thus reduced to providing a list of the relevant accession numbers.

We consider the situation here in which there are n binary attributes on which searching is being performed. Moreover, we consider queries which request those records which do possess certain attributes; the extension to the case in which we further stipulate that the records not have certain other attributes is not essentially more difficult. In a typical retrieval system, queries are relatively simple, in that they involve relatively few of the attributes. Hence, we first consider the case where partial match queries on up to t attributes must be supported, but queries on more than t attributes need not be.

The usual *inverted file system* creates a list of accession numbers for each attribute, and intersects these lists to reply to a partial match query. This requires the examination of very many accession numbers which do not form part of the final answer. At the other extreme, an *extended* inverted file system creates (in advance) a "bucket" of accession numbers for each partial match query. Redundancy is incurred in this scheme, but can be limited by only placing an accession number in a bucket when the partial match query is a maximal query which matches the record. The redundancy in storage pays off in retrieval, because in this scenario only the relevant accession numbers will be examined. The impracticality of this approach arises from the very large number of buckets required, and a correspondingly large requirement for redundancy.

A compromise solution is to amalgamate many possible queries into a single bucket. Each bucket remains associated with a subset of the attributes, but may now contain information about many maximal partial match queries. The essential feature of the bucket subsets is that each query subset be contained in at least one bucket subset.

Now we are in a position to introduce combinatorial filing schemes, as introduced in Bucholz (1963) and developed by Abraham et al. (1968) and Ray-Chaudhuri (1968). We need one further definition: a t -covering is a set system (X, \mathcal{B}) in which every t -subset appears in at least one block in \mathcal{B} .

Let $A = \{a_1, \dots, a_n\}$ be a set of attributes. Let (A, \mathcal{B}) be a t -covering, and write $\mathcal{B} = \{B_1, \dots, B_m\}$. Each B_i has an associated list, or *bucket* M_i . Not all subsets of A appear in blocks of \mathcal{B} , but we are guaranteed that all t' -subsets with $t' \leq t$ are. A subset $A' \subset A$ which does appear may appear in many blocks; we write $f(A') = i$ if the "first" block in which the subset A' appears is B_i . Now many subsets are associated with bucket M_i , and hence we partition this bucket into *subbuckets*; in particular, for each $A' \subseteq A$ with $f(A') = i$, we form a subbucket $M_{i,A'}$.

To enter a new record with attributes R , we place its accession number in subbucket $M_{i,A'}$ provided that $R \cap B_i = A'$ and $f(A') = i$. Each accession number thus appears in at most one subbucket of each bucket, but may appear in many different buckets. To answer a partial match query Q , we determine $i = f(Q)$ and only examine bucket M_i . The relevant accession numbers are then listed, each exactly once, by

catenating all of the subbuckets $M_{i,A'}$ with $Q \subseteq A'$.

If only one bucket is used, this scheme reduces essentially to extended inverted filing. In fact, within each bucket, the scheme is like extended inverted filing, with one important difference. Subbuckets $M_{i,A'}$ exist even for sets A' which are too large to be partial match queries themselves; this eliminates redundancy within a bucket. The main advantage of first partitioning into buckets in this way is that the filing problems remaining within a bucket are intended to be of manageable size.

Two competing goals affect the selection of t -covering to be used. First, the redundancy incurred by storing accession numbers in many buckets dictates that the t -covering should have few blocks; intuitively, fewer blocks lead to less redundancy. Second, larger blocks lead to more subbuckets per bucket, and hence leave larger filing problems within a bucket; intuitively, one prefers smaller blocks. The tradeoff between having few larger blocks or many smaller blocks is very application dependent. When $t = 2$, Ray-Chaudhuri (1968) observes that the first goal suggests the use of projective planes; Koch (1969) develops closely related t -coverings for the cases when planes are not known to exist. If the second goal is taken into account, block designs with small blocksize are preferable (Ray-Chaudhuri (1968)). When $t > 2$, the designs to be used are not as readily available, especially in view of the requirement that the index be 1. For $t \leq 5$, many suitable designs are known to exist, but as noted earlier, existence is far from settled in general.

A most profitable direction to extend this research has been considered by Bose and Koch (1969) and Ray-Chaudhuri (1968). On each bucket, we can develop a second combinatorial filing scheme, and thereby develop an overall method which is multi-stage. To do this, on each block of a $t-(v,k,1)$ design, we place a copy of a $t-(k,k',1)$ design. The operation of the filing scheme is to first find the relevant block of the $t-(v,k,1)$ design, and then within that block find the relevant block of the $t-(k,k',1)$ design; this could naturally be repeated to form a filing scheme with any desired number of stages. In practical terms, the lack of known Steiner systems with large t and k limits the usefulness of this idea, however. Even when appropriate systems are known, one might argue that the result is just a $t-(v,k',1)$ design. Of course, it is such a design, but has the advantage that we need not search all blocks of the design in order to locate the relevant bucket. One would employ two mappings here, one to locate the relevant block of the $t-(v,k,1)$ design, and the second to locate within that subset the relevant block of the $t-(k,k',1)$ design.

A second profitable direction is to generalize the scheme to handle multiple valued attributes. The extension of the design-theoretic approach to this problem has been studied by many authors, notably Bose and Koch (1969), Bose et al. (1969), Ghosh and Abraham (1968), Chow (1969), Takahashi (1973) and Berman (1976).

Rivest (1974a,1976) points out two major practical limitations to the combinatorial filing schemes. The first is a lack of available designs for larger t , which makes the schemes impractical for $t > 3$, at least at present. The second is the large redundancy introduced by storing accession numbers many times; while this may be quite acceptable for small files, one would require a very large difference in the sizes of records and their accession numbers before the storage for buckets would be less than storage for the file itself. Nevertheless, combinatorial filing schemes remain useful when the cost of retrieving a record from the secondary storage is so large that one is unwilling to retrieve any record which may prove irrelevant to the query at hand. In the absence of such a prohibitive cost, however, we need only ensure that "most" of the records retrieved prove relevant; Rivest's scheme, which we explore next, has this property.

Rivest (1976) considers partial match queries of any size on a file with n binary attributes; a query specifies records which do possess certain properties, do not possess certain others, and may or may not possess the remainder. In this case, records are placed in buckets; however, here the buckets partition the records, i.e. no redundancy is permitted. A record R is placed in a bucket M_i by evaluating a hash function h ; if $h(R) = i$, R is placed in M_i . The hash function is therefore a function which partitions all possible records into b buckets M_1, \dots, M_b . To answer a partial match query Q , we determine (in a manner as yet unspecified) all buckets which could contain a record matching Q , and then linearly search all of the selected buckets. (Notice that if accession numbers rather than actual records are stored, this means we must access the secondary storage to retrieve all of the records in these buckets.)

The essential ingredient here is the selection of the hash function. It must have two properties. We must be able to easily determine whether records in a given bucket could possibly match a given query. In addition, we want to examine as few buckets as possible (either on average or in the worst case). Notice that these decisions are not affected by the file itself. Rivest's main theorem here shows that if we have $b = 2^w$ buckets, to minimize the average number of buckets examined we choose a function which hashes a group of records to the same bucket if they are "close" in the following sense. For bucket M_i , there exist sets S_i^0 and S_i^1 for which all records which have attributes in S_i^0 set to 0 and attributes in S_i^1 set to 1, and no others, are hashed to M_i . Moreover, the number of specified attributes $|S_i^0 \cup S_i^1|$ is w . Hence the set of records hashed to bucket M_i can be easily encoded as an n -vector with entries $\{0,1,*\}$ containing w digits and $n-w$ *'s; we call these vectors *signatures* of the buckets. The asterisks denote "don't care" positions.

An easy example when $n > \log_2 b$ simply uses the first $\log_2 b$ bits of the record to determine the bucket. While this easy method has optimal average case performance, in the worst case it may require the examination of all buckets. To illustrate this, the simple scheme with $w = 2$, $n = 3$ and $b = 4$ uses signatures:

0	0	*
0	1	*
1	0	*
1	1	*

The query (* * 1) examines all buckets, while the query (0 * *) only examines the first two. A better worst case is achieved by using signatures:

0	0	*
0	1	*
1	*	0
1	*	1

The improvement results from distributing the *'s more uniformly across the columns. What yields the best worst-case complexity? Rivest (1974b,1976) addresses this question by considering a novel type of designs. An *associative block design* $ABD(n,w)$ is a $2^w \times n$ array with entries from $\{0,1,*\}$ so that

- (1) each row has w digits and $n-w$ *'s,
- (2) for every pair of rows, there is a column in which they contain different digits, and
- (3) every column contains the same number, $2^w(n-w)/n$, of *'s.

Conditions (1) and (2) ensure that the signatures (=rows) form a partition of the file, which when used as a hash function delivers optimal average case performance. Condition (3) is designed to ensure that worst-case performance is also good. Rivest explores the existence of *ABDs* for many parameter sets, and employs a number of product constructions borrowed from more standard design existence problems. Brouwer (1976) establishes the existence of many *ABDs*, but existence remains far from settled. The lack of appropriate *ABDs* led Rivest (1976) to explore various relaxations of *ABDs*, by allowing signatures with more than w bits specified, allowing more than one signature per bucket, and allowing redundancy. In the latter case, we obtain what we might call "associative coverings".

Burkhard (1976a, 1976b) also considers the development of designs when no *ABD* is available; he employs a more general class of designs, which drops the third requirement in the definition of associative block designs. He develops a simple recursive construction for this more general class of designs; however, Burkhard's file designs seem to be only peripherally related to block designs.

In the area of filing schemes and partial match retrieval, the balanced appearance of subsets once again plays a key role; for filing schemes, the balancing minimizes redundancy, while for partial match retrieval, the balancing underlies good worst-case retrieval.

7. Sorting in Rounds

In this section, we consider some applications of design theory to the problem of sorting in rounds. Given n distinct elements in some order, the linear order of the elements is to be established through binary comparisons. Each comparison (question) establishes the relative order (answer) for a pair of elements, and having determined the relative ordering of several pairs of elements, additional orderings may be deduced by implication. For example, if $x > y$, $y > z$ and $z > w$, we can deduce that $x > w$. This is an example of a 3-step implication; a 2-step implication is a *direct implication*. Given n elements and a positive integer k , the problem of *sorting in rounds* involves determining the complete order of the elements in k rounds. In each round, a number of questions is answered simultaneously, and the questions asked within one round can be formulated on the basis of answers from all previous rounds; the object is to ask as few questions as necessary. One application of the problem is in situations where comparisons are made via correspondence, for example, in testing consumer preferences; see Häggkvist and Hell (1982).

The connection between sorting in rounds and parallel sorting is immediate: if r questions are asked in a round, then by assigning one comparison to each of r processors, each round can be completed in one time unit. Under the comparison-cost model, the cost is determined to be the number of comparisons made (number of processors required); work involved in formulating questions to be asked in succeeding rounds, communication between processors, data storage, and so on, is not considered. It is well known that $O(n \log n)$ comparisons are necessary and sufficient to sort n elements using standard binary sorting; this corresponds to asking one question per round, using $k = O(n \log n)$ rounds. At the other extreme, if sorting is to be accomplished in a single round, all $\binom{n}{2}$ questions must be asked. The cases of interest are hence for $1 < k < n \log n$.

Consider sorting in two rounds. Let V be the set to be sorted. A question (comparison) is an unordered pair of elements from V . Let E be the set of questions asked in the first round, and observe that $G = (V, E)$ is an undirected graph. Now an answer for the question $\{x, y\}$ is either (x, y) or (y, x) , according to whether $x > y$ or not. Replacing each $e = \{x, y\} \in E$ by the answer (x, y) or (y, x) transforms G into an *oriented graph*. In fact, the sets of possible answers correspond precisely to the *acyclic orientations* of G .

The [d -step] *transitive closure* of an oriented graph G is the oriented graph with the vertex set of G and an arc from vertex v to vertex w if and only if there is a path [of length at most d] from v to w in G . A 2-round sorting algorithm using r processors to sort n elements corresponds to a graph G with n vertices and at most r edges, such that the transitive closure G^c of any acyclic orientation of G contains at least $\binom{n}{2} - r$ arcs; the edges of G correspond to the questions asked in the first round, and the (at most r) edges *not* in G^c are the questions which remain to be asked in the

second round.

Häggkvist and Hell (1981) establish the existence of a 2-round sorting algorithm using $O(n^{5/3} \log n)$ processors. Unfortunately, their proof is non-constructive, and relies on the existence of graphs with the specified property. Informally, what is required is a sparse graph G with the property that the transitive closure of every acyclic orientation of G is dense. In an attempt to actually construct such a graph (i.e. explicitly exhibit an efficient 2-round sorting algorithm), they make use of a known class of Steiner systems: for all $n \equiv 1$ or $10 \pmod{90}$ and sufficiently large, $(v, k, \lambda) = (n, 10, 1)$ designs can be employed, yielding an $O((13/30)(n^2 - n))$ 2-round sorting algorithm. However, such an algorithm is ensured only for values of n which are too large for practical application; an improved existence result for the required designs would likely make the scheme practical.

Alon (1985) employs the points and hyperplanes of the projective geometry of dimension 4, $PG(4, q)$, to construct explicit algorithms for sorting n elements in two rounds using only direct implications. As noted above, a graph G corresponding to an efficient 2-round sorting algorithm using direct implications only is a sparse graph G for which the 2-step transitive closure of every acyclic orientation of G is dense. Now the points of $PG(m, q)$ can be put into one-to-one correspondence with the hyperplanes, with say point x_H corresponding to hyperplane H , such that the following graph G is well-defined. G has one vertex v_H for each point x_H of $PG(m, q)$, with v_{H_1} joined to v_{H_2} in G if and only if $x_{H_1} \subset H_2$. Then for $m = 4$, from the above parameters, it follows that G has $(1 + o(1))n^{7/4}$ edges, for $n \rightarrow \infty$ with m fixed, and it can be proven that the 2-step transitive closure of any acyclic orientation of G has at most $O(n^{7/4})$ edges missing. This gives an algorithm for sorting $n = (q^5 - 1)/(q - 1)$ elements, for any prime or prime power q , in 2 rounds. This uses only direct implications. Only $r = O(n^{7/4})$ processors are required, considerably improving on the algorithm exhibited by Häggkvist and Hell. The algorithm is evidently close to optimal, since a lower bound establishes $r = \Omega(n^{5/3})$. If the number n of elements to be sorted does not have the form $(q^5 - 1)/(q - 1)$, then "virtual elements" can be added and handled appropriately, without increasing the complexity of the sorting scheme.

In an earlier paper, Bollobás and Rosenfeld (1981) use the same construction for $m = 2$ (i.e. projective planes) in their analysis of the related problem of "almost-sorting" in a single round: they consider the number of comparisons required in one round such that "almost all" comparisons can be deduced from the results of these by direct implications. Alon (1985) relates that using the points and hyperplanes of the projective geometry of dimension $m = 3$, Pippenger has produced an algorithm with complexity $O(n^{26/15})$ for sorting n elements in two rounds using implications of arbitrary length; moreover, combining his own work with a construction due to Häggkvist and Hell (1982), explicit k -round sorting algorithms which are more efficient than the best previously known can be constructed for all constant $k \geq 4$. A recent paper

discussing the problem of sorting in rounds under the parallel computation model, containing further results on the complexity of parallel sorting, is given by Azar and Vishkin (1987); see also Pippenger (1987).

8. Probabilistic and Deterministic Algorithms

We now turn our attention to the use of block designs in the design and analysis of algorithms. First, we discuss some connections between block designs and the problem of finding roots of polynomials over finite fields; we then mention the use of block designs in constructing an algorithm to solve the maximal independent set problem.

Given a finite field $F = GF(q)$, where q is a prime or prime power, and a polynomial $f(x) \in F[x]$ which factors into distinct linear factors over F , the *root-finding problem* is to determine the roots of $f(x)$ in F . The problem is important in algebraic computation and arises in the areas of algebraic coding theory (see MacWilliams and Sloane (1978)) and cryptography (see Zierler (1974) and Odlyzko (1985)), among others. For example, in coding theory, the roots of the error-locator polynomial determine the coordinate positions at which errors in a received vector have occurred. Some relations between block designs and root-finding as explored by van Oorschot and Vanstone (1987) are outlined below, particularly the inherent appearance of block designs in two known root-finding algorithms.

A probabilistic root-finding algorithm for q odd was introduced by Berlekamp (1970) and further discussed by Rabin (1980). Given $f(x)$, select a random $c \in F$ and compute

$$\gcd(f(x), (x+c)^{(q-1)/2} - 1) \quad (1)$$

If the gcd is trivial, repeat the computation with another random $c \in F$. Otherwise, a factorization $f(x) = f_1(x)f_2(x)$ has been discovered, and the method is applied recursively to $f_1(x)$ and $f_2(x)$ until the linear factors of $f(x)$ are found.

For $c = 0$, (1) separates the roots of $f(x)$ which are quadratic residues from those which are not, and in general separates roots ρ_i and ρ_j only if $\rho_i + c$ and $\rho_j + c$ are not both quadratic residues or both quadratic nonresidues. To relate this algorithm to block designs, we require some notation. Let R be the set (block) of quadratic residues (even powers of a generator) in F , and let N be the set of quadratic nonresidues. Define $dev_F(R, N)$, the *development* of $\{R, N\}$ by F , to be the set of q near-resolution classes $\{R+\beta, N+\beta\}$, $\beta \in F$, where for example $R+\beta = \{r+\beta: r \in R\}$. It is then well-known that $dev_F(R, N)$ is a $(q, (q-1)/2, (q-3)/2)$ block design. This implies that a pair of roots $\{\rho_1, \rho_2\}$ of $f(x)$ appears together in precisely $(q-3)/2$ of the q classes. Selecting a random $c \in F$ for (1) essentially corresponds to selecting a random class in this design; two roots remain unseparated if they appear in the same block of this class. (To be precise, the corresponding design is actually $dev_F(R, N \cup \{0\})$.) The recognition of this underlying design provides one method for analyzing this root-finding algorithm. In particular, the probability p that a given pair of roots is

separated via a random choice of c in (1) can be precisely determined: for $q \equiv 3$ modulo 4, $p = (q+1)/2q$, and for $q \equiv 1$ modulo 4, $p = (q+3)/2q$ or $(q-1)/2q$, depending on the quadratic nature of the difference of the roots.

This analysis technique using block designs is particularly convenient in that it generalizes to handle the modified version of this root-finding algorithm which makes use of a multiplicative subgroup of $F \setminus \{0\}$ of index n (in place of the quadratic residues) and its cosets, and can also be used to establish the exact probability that *three* roots of $f(x)$ remain unseparated by (1). In addition, the underlying designs can be used to establish bounds on the size of *factorizing subsets* as introduced by Camion (1983), in relation to the existence of a deterministic version of a related probabilistic general factorization algorithm.

A second algorithm introduced by Berlekamp (1970), for finding roots over extension fields $GF(q^m)$, $m > 1$, makes use of the following result. Let α be the root of an irreducible polynomial of degree m over $GF(q)$, and let $tr(x) = \sum_{i=0}^{m-1} x^{q^i}$ be the standard *trace* function over finite fields. Then the set of greatest common divisor computations

$$\gcd(f(x), tr(\alpha^j x) - \beta), \quad (2)$$

where $j = 0, 1, \dots, m-1$ and β runs over all elements of $GF(q)$, separates all roots of $f(x)$. This gives a deterministic root-finding algorithm. Furthermore, for fixed j and β ranging over $GF(q)$, the exact probability that (2) separates a pair of roots of $f(x)$ can be determined in a manner similar to that used above. Here, it can be established that the underlying design is given by the points and hyperplanes of the affine geometry $AG(m, q)$, a $(q^m, q^{m-1}, (q^{m-1}-1)/(q-1))$ design with replication number $(q^m-1)/(q-1)$. Again, selecting a random j corresponds to selecting a random resolution class of this design, and the probability that such a class contains a given pair of roots among one of its blocks (and hence fails to separate that pair) is $\lambda/r = (q^{m-1}-1)/(q^m-1) < 1/q$. This analysis further motivates a generalization of this root-finding algorithm employing the block design defined by the points and subspaces of dimension $l < m-1$ of $AG(m, q)$, i.e. $AG_l(m, q)$; see Beutelspacher et al. (1987).

We now discuss another use of block designs in algorithm design and analysis: the establishment of a complexity result for the maximal independent set problem. Given a graph $G = (V, E)$, an *independent set* in G is a subset $I \subseteq V$ such that $u, v \in I$ implies $\{u, v\} \notin E$. A *maximal independent set* is an independent set that is not a subset of any larger independent set in G ; the *maximal independent set problem* is to produce a maximal independent set I in G .

For $S \subseteq V$, the *neighbourhood* of S is $N(S) = \{w \in V \mid \text{for some } u \in S, \{u, w\} \in E\}$, and $N_K(S) = N(S) \cap K$. S is then an independent set if $S \cap N(S) = \emptyset$, and S is a maximal independent set if furthermore $S \cup N(S) = V$. A simple sequential algorithm to find a maximal independent set I starts with $I = \emptyset$, and then for each vertex $v_i \in V$, $1 \leq i \leq n$, simply adds v_i to the set I if $v_i \notin N(I)$.

Karp and Wigderson (1985) present a randomized parallel algorithm for the maximal independent set problem that runs in expected time $O((\log n)^3)$ using $O(n^2)$ processors, and a deterministic parallel algorithm that runs in time $O((\log n)^4)$ with $O(n^3/(\log n)^3)$ processors, where $n = |V|$. An outline of the algorithm follows. Start with sets $I = \emptyset$ and $H = V$, and proceeding until $H = \emptyset$, repeatedly

- (i) find an independent set S in the subgraph induced by H
- (ii) update I to $I \cup S$
- (iii) delete from H all vertices in $S \cup N_H(S)$

At termination, I is a maximal independent set. Step (i) in this algorithm admits parallel computation.

Karp and Wigderson show how to find S in step (i) in time $O((\log n)^2)$ such that

$$|S \cup N_H(S)| = \Omega(|H| / \log |H|), \quad (3)$$

leading to a requirement of at most $O((\log n)^2)$ iterations. In greater detail, given a subset $T \subseteq H$, if one vertex from each edge in the subgraph induced by T is deleted from T , then the remaining vertices in T form an independent set S . For *good* sets $T \subseteq H$, i.e. sets T carefully selected as outlined below, the resulting S satisfies (3).

For $K \subseteq H \subseteq V$, let d be the maximum degree of any vertex in the subgraph induced by K , and call a vertex $u \in K$ *heavy* if the degree of u in K is at least $d/2$. Karp and Wigderson establish that if $M \subseteq K$ is a set of heavy vertices in K , then for $t = m/8d$, where $m = |M|$, there exist t -sets $T \subseteq M$ such that independent sets S , constructed from T as in the preceding paragraph, satisfy $|S \cup N_H(S)| = \Omega(m)$. To establish (3), given a set $H \subseteq V$, they then show how to construct a subset $K \subseteq H$ which contains a set $M \subseteq K$ of heavy vertices with $m = \Omega(|H| / \log |H|)$. Then with K , M and t fixed, what remains is to find, among all t -sets of M , a good set T .

At this point, the randomized version of the algorithm proceeds by randomly selecting subsets $T \subseteq M$ of cardinality t until a good set T is found. The deterministic version systematically employs a particular class of Hadamard designs to replace the random selections by a deterministic (parallel) search procedure.

The properties that make block designs useful in the deterministic algorithm are precisely the constant index and existence of a (unique) replication number (for all points). Although the randomized version is strongly recommended if the algorithm is to be programmed, the use of block designs is required to establish the deterministic complexity result. Karp and Wigderson (1985) state that their main contribution is "introducing combinatorial design theory as an algorithmic technique". They note that the use of block designs, to replace random sampling by deterministic sampling, is precisely the reason that block designs were originally studied in agriculture and statistics. They go on to say

We believe that combinatorial designs will find many applications in the design of efficient deterministic algorithms, and particularly in parallel algorithms, where they seem to fit so

naturally.

Karp, Upfal and Wigderson (1985) consider a generalization of the maximal independent set problem for graphs. An *independence system* is a set X , together with a set \mathcal{C} of subsets of X ; whenever $C \in \mathcal{C}$ and $C' \subset C$, $C' \in \mathcal{C}$. For example, the set \mathcal{C} of independent sets in a graph forms an independence system on the set of its vertices. Karp, Upfal and Wigderson consider the following problem for an n -element independence system (X, \mathcal{C}) : given an oracle for deciding whether $C \in \mathcal{C}$ for $C \subseteq X$, find a *maximal set* in \mathcal{C} . They give a randomized parallel algorithm for this problem in $O(\sqrt{n})$ time; once again, designs (in this case, t -(v, k, λ) packings) are used to replace the random sampling by deterministic sampling. For a special class of independence systems (matroids), this gives a deterministic parallel algorithm in $O(\sqrt{n})$ time. Restricting further still, they consider the independence system obtained by considering all subsets of the edge set of a graph which contain no cycle (this independence system is the "graphic matroid"). Using t -(v, k, λ) packings, they obtain an $O(\log n)$ deterministic algorithm for finding a maximal set. In concrete terms, suppose that we are to find a spanning tree in a connected graph G , but are not allowed to "see" G ; we can only ask an oracle whether a subset of edges contains a cycle or not. Their algorithm using packings finds a spanning tree of G in $O(\log n)$ time, using a subexponential number of processors.

9. Lower Bounds for Algorithms

In the previous section, the use of designs in developing algorithms was illustrated; in this section, we explore an application of designs to establishing a lower bound on the performance of an algorithm. We consider the *set covering* problem: given a set system (V, \mathcal{B}) , find a set $X \subseteq V$ which is a set cover, i.e. $X \cap B \neq \emptyset$ for all $B \in \mathcal{B}$, so that $|X|$ is minimum. This problem arises, for example, in choosing sites in a radio broadcast network to serve as repeater stations (Van Slyke (1982)). The usual method by which set covering problems are solved is to use an "integer programming" formulation with "implicit enumeration"; Fulkerson, Nemhauser and Trotter (1974) describe these methods in detail. In order to avoid introducing new terminology, we recast their method in the context of set systems.

The algorithm operates by maintaining a list of candidate partial solutions, along with a size of the smallest set cover found thus far. At each step of the algorithm, a partial solution is eliminated from the list, if it can be seen not to lead to a set cover smaller than the current best, or smaller than one arising from a different partial solution.

To be more precise, we view a partial solution as a partition (I, O, U) of V ; I contains elements which we have decided to place in the set cover, O contains elements which we have decided not to place in the set cover, and U contains elements about which we must yet make a decision. We let $\mathcal{R}(I)$ be the set of blocks not covered by

I . Initially, we take $(V, \emptyset, \emptyset)$ as the smallest known solution, and the initial set of partial solutions is $\{(\emptyset, \emptyset, V)\}$. Given a list of partial solutions, we employ three phases: augmentation, elimination, and branching. In *augmentation*, we check whether, for any partial solution (I, O, U) , there is a $u \in U$ and $B \in \mathcal{B}$ for which $B \cap (I \cup U) = \{u\}$. If so, then u must be placed in the set cover in order to cover the set B ; hence we replace (I, O, U) by $(I \cup \{u\}, O, U \setminus \{u\})$. *Elimination* of partial solutions is done in two ways. Elimination *by dominance* is based on the observation that for two partial solutions (I, O, U) and (I', O', U') with $\mathcal{R}(I') \subseteq \mathcal{R}(I)$, $|I'| \leq |I|$ and $I \neq I'$ a set cover extending (I, O, U) can never be better than one extending (I', O', U') ; thus (I, O, U) can be eliminated from the list of partial solutions. To describe elimination by fathoming, we recast the set covering problem as the determination of weights from $\{0, 1\}$ for each element so that every $B \in \mathcal{B}$ has elements whose weight totals at least one. In a partial solution, I contains elements whose weight is fixed at 1, O those whose weight is fixed at 0, and U those whose weight is yet to be determined. Elimination *by fathoming* then determines the best (non-negative) *fractional* assignment of weights to the elements of U , so that each set of \mathcal{B} obtains total weight at least one (i.e., it solves a "linear programming relaxation"). If the total weight assigned, even allowing fractional weights, is at least the weight of the current best set cover, the partial solution can be eliminated from consideration.

Once each partial solution is augmented as much as possible, and all those which can be seen to lead to unusable solutions are eliminated, we perform a *branching*; that is, for some partial solution (I, O, U) and some $u \in U$, we replace this partial solution by the two partial solutions $(I \cup \{u\}, O, U \setminus \{u\})$ and $(I, O \cup \{u\}, U \setminus \{u\})$.

A lower bound on the complexity of such an algorithm is obtained by examining the number of partial configurations which it considers. Avis (1980) observes that for set systems (V, \mathcal{B}) which are Steiner triple systems $((v, 3, 1)$ designs), the algorithm described can take exponential time. In particular, it performs poorly on Steiner triple systems whose smallest set cover is large; de Brandes and Rödl (1984) have demonstrated the existence of Steiner triple systems whose minimum set cover has size $v - O(v^{1/2})$.

To develop the lower bound, form a binary computation tree whose nodes are the partial solutions examined; the two children of an interior node in this computation tree are the two partial solutions obtained by branching. Avis (1980) shows that this binary tree is full to a depth of at least $q = \lfloor \sqrt{2v/3} \rfloor$. To see this, consider a partial solution (I, O, U) . Let j_0 be the number of elements in O (placed there by branching). Let j_1 be the number of elements placed in I by branching, and let j_2 be the number placed in I by augmentation.

In a Steiner triple system, no two blocks intersect in more than one element. Hence every pair of elements added to O can be responsible for the addition of at most one element to I by augmentation. Thus we have $j_2 \leq \frac{1}{2}j_0(j_0 - 1)$. Now if $j_0 + j_1 \leq q$,

a fractional solution simply assigns weight $\frac{1}{2}$ to all elements of U ; the weight of the "solution" is then at most $2v/3$. Since the weight of the best integer solution is $v - cv^{\frac{1}{2}}$, these partial solutions are not eliminated by fathoming. They are also not eliminated by dominance, for the following reason. If (I, O, U) is eliminated by (I', O', U') , then $\mathcal{R}(I') \subset \mathcal{R}(I)$ and there is some $x \in I \setminus I'$. But then each of the blocks containing x must intersect I' in the partial solution (I', O', U') ; this requires $|I'| \geq r = (v-1)/2$, which is a contradiction for $j_0 + j_1 \leq q$. Thus there is no elimination of partial solutions by dominance.

The key feature of the designs used is the small intersection of blocks; however, the other essential observation is that Steiner triple systems exist having large minimum set covers.

More recently, properties of Hadamard designs have been used to establish lower bounds in the theory of probabilistic communication complexity. Chor and Goldreich (1985) obtain a linear lower bound on the worst-case probabilistic communication complexity for computing functions which are representable by Hadamard designs. To do this, they exploit a remarkable property of Hadamard designs: in a Hadamard design of order v , let X be a set of i elements and Y be a set of j blocks. Let $c(X, Y)$ be the total number of occurrences of elements of X in blocks of Y . Then $\frac{1}{2}(ij - \sqrt{ijv}) \leq c(X, Y) \leq \frac{1}{2}(ij + \sqrt{ijv})$ (see Frankl, Rödl and Wilson (1987)). This (relatively) uniform distribution of elements in blocks leads to the lower bound; we do not attempt to sketch the details here, but refer the reader to Chor and Goldreich (1985) and to Babai, Frankl and Simon (1986).

10. Interconnection Networks

In this section, we examine interconnection strategies for computer networks. The generic *interconnection problem* is to establish communication paths between *input nodes* V_I and *output nodes* V_O ; we allow the case that the input nodes and output nodes are the same. *Switch nodes* V_S may be used; these simply relay a message from one communications channel to another. Our task is to connect the nodes in $V = V_I \cup V_O \cup V_S$ using some communication medium. Two main media are bidirectional point-to-point *links*, and (multipoint) *buses*. Ignoring implementation issues, a link is just a 2-subset of nodes, while a bus is a k -subset of nodes, for $k \geq 2$. We can therefore view a network as a set system (V, \mathcal{B}) , where \mathcal{B} is the collection of links or buses; for simplicity, we treat links as a special case of buses.

The physical realization of a network imposes some basic constraints:

1. The number of buses should be small.
2. Each node should connect to a small number of buses.
3. The number of switch nodes should be small.

4. Each bus should involve a small number of nodes.

The first three constraints address the construction cost, while the fourth is to ensure that each bus carries an acceptably small amount of traffic. The network must also be *connecting*, in that there is a communication path from each input node to each output node. The *distance* from an input node to an output node is the number of buses in the shortest communication path connecting them. The *diameter* or *depth* is the maximum distance from an input to an output. To ensure small delay in communication, we also require:

5. The diameter should be small.

Naturally, these five goals are conflicting, and any network design is a compromise solution. Let us consider first a network (V, \mathcal{B}) with diameter one, and $V = V_I = V_O$. In this case, no switch nodes are required. For the diameter to be one, every 2-subset of V must appear in some block of \mathcal{B} ; hence (V, \mathcal{B}) is a $2-(|V|, K, 1)$ covering. Mickunas (1980) observes that minimizing the number of buses, and also the number of buses meeting each node, we take (V, \mathcal{B}) to be a projective plane. If there is no plane on $|V|$ elements, Mickunas suggests simply omitting points from a larger plane to obtain a pairwise balanced design which meets the criteria.

Bermond, Bond and Saelé (1984) and Bermond and Bond (1986) consider the problem of building networks with specified maximum bus size k , in order to meet the requirement for small buses. Such a network is a covering with small block sizes; the minimum number of blocks is achieved by a $2-(v, k, 1)$ design when there is such a design on the required number of elements.

Thus far, we have considered networks in which any single input node can reach any single output node (when inputs and outputs are equal, we require that any pair of nodes can be connected by a path). More generally, we may require that the n input nodes can simultaneously communicate with the n output nodes, given a specified mapping of inputs to outputs. This requires *disjoint* communication paths, which share no common bus or intermediate node. A good example of this situation arises in the design of shifting networks.

A *barrel shifter* is a network whose nodes are $\{0, 1, \dots, n-1\}$, the integers modulo n . Given a shift distance s , $1 \leq s < n$, every node must transfer a value to the node whose label is s larger; more precisely, for each $0 \leq i < n$, node i must establish a connection to node $i+s$ (modulo n), and all n communication paths are to be disjoint. Kilian, Kipnis and Leiserson (1987) develop a barrel shifter which has diameter one; as they remark, when implemented in VLSI, this means that the shift is accomplished in a single clock tick.

As we have seen, a network of diameter one is a $2-(n, K, 1)$ covering; if we require in addition that each node i has a (disjoint) path to node $i+s$ (modulo n), the n pairs from the set $D_s = \{(i, i+s) \bmod n : 0 \leq i < n\}$ must appear in n distinct blocks. At first, this seems to be a complicated requirement, but a widely studied class of designs

always has the desired property; we introduce them here. A set system (V, \mathcal{B}) with $V = \{0, 1, \dots, n-1\}$ is *cyclic* if, whenever $\{b_1, \dots, b_k\} \in \mathcal{B}$, $\{b_1+1, \dots, b_k+1\} \in \mathcal{B}$ (arithmetic modulo n is used here). Colbourn and Mathon (1980) survey results on cyclic designs; we introduce only those features which are essential for our purposes. The *orbit* $\mathcal{O}(B)$ of a block B is the set $\{B+s \pmod n : 0 \leq s < n\}$; it is *full* when $|\mathcal{O}(B)| = n$. When all orbits are full, the set system is *full-cyclic*. It is easy to see that the pairs of D_s appear in at least n distinct blocks of a full-cyclic covering.

Any full-cyclic covering can then be used to design a barrel shifter. Each node finds the first orbit in which $\{0, s\}$ appears, say in block B . Node x now writes its value to the bus $B+x \pmod n$, and reads its value from the bus $B+x-s \pmod n$. In this way each node x reads the value node $x-s \pmod n$ wrote, and each communication path corresponds to a unique block in the orbit. Kilian, Kipnis and Leiserson (1987) observe that to minimize the total number of buses and the number of buses incident at a node, the covering chosen here is a *cyclic projective plane* (i.e. projective plane which is cyclic).

The actual operation of a barrel shifter based on a cyclic projective plane is remarkably simple. To see this, we consider the structure of cyclic projective planes. Since there are only n blocks, any two blocks B_1, B_2 satisfy $B_1 \equiv B_2 + s \pmod n$ for some $0 \leq s < n$. Consider a single block $B = \{b_1, \dots, b_k\}$. Now for each element d , $1 \leq d < n$, $\{0, d\}$ appears in exactly one block. Hence B must contain exactly two elements b_i, b_j for which $b_j - b_i \equiv d \pmod n$. Every d , $1 \leq d < n$, is the difference of two elements of B ; such a set B is a *difference set* for $\{0, 1, \dots, n-1\}$.

Using the difference set representation of the cyclic projective plane, the operation of a barrel shifter is straightforward. To shift a distance of s , each node finds the two elements b_i, b_j in the difference set with $b_j - b_i \equiv s \pmod n$. Node x then writes onto bus $x+b_i$, and reads from bus $x+b_j$.

When no cyclic projective plane on n elements exists, this very simple control logic can be retained nonetheless. To do this, note that this scheme requires only a set which covers all differences from 1 to $n-1$; hence we can use a *difference cover*, in which each d , $1 \leq d < n$, is the difference of at least one pair of elements. Babai and Erdős (1982) establish the existence of 'small' difference covers, and Kilian, Kipnis and Leiserson (1987) observe that they produce optimal barrel shifters of depth one. They also use difference covers to design 'permutation architectures', which realize permutations other than just cyclic shifts.

We now consider an even stronger connection property of networks. An *n-superconcentrator* is a network with n inputs and n outputs in which disjoint communication paths can be established from the inputs to the outputs in *any* of the $n!$ possible orderings. We restrict superconcentrators to have only links, and no larger buses. A superconcentrator of depth one requires all n^2 connections (i.e. each input connected to each output); hence superconcentrators of depth greater than one are of

interest. Nevertheless, superconcentrators are typically constructed using special types of depth-one networks in which every set of inputs is directly connected to a relatively large set of outputs (see, for example, Chung (1979)). More formally, a network $(V_I \cup V_O, E)$ with $V_I \cap V_O = \emptyset$ is a (n, α, β) -expander if every set of α inputs is directly connected to at least β output nodes.

Any depth-one network with $V_I \cap V_O = \emptyset$ can be equivalently written as a set system (V_I, \mathcal{B}) , where $\mathcal{B} = \{\{v_i : \{v_i, v_o\} \in E\} : v_o \in V_O\}$. In this setting, an (n, α, β) -expander is a set system with n elements and n blocks, so that every set of α elements intersects at least β of the blocks. Intuitively, β is largest when the blocks intersect each other as little as possible. At the same time, however, for β to be large, each element must appear in a large number of blocks. As one would expect, to maximize the expansion, we choose to balance the block sizes, and balance the sizes of block intersections. Hence we consider symmetric designs.

Alon (1985) proves that one class of symmetric designs, obtained from the points and hyperplanes of the projective geometries $PG(d, q)$, provides good expansion properties. More precisely, he shows that in this design from $PG(d, q)$ on n elements, every set of α elements intersects $\beta \geq (\alpha n)/(\alpha + q - 1)$ blocks. Hence for all $\alpha = o(n)$, $\alpha = o(\beta)$; such a network is termed *highly expanding*. Moreover, Alon remarks that these expanders have essentially the smallest number of links of any network with equivalent expansion properties. Using projective geometries for expanders, Alon establishes the existence of n -superconcentrators of depth three with $O(n^{4/3})$ links; we refer the reader to Alon's paper for further uses of the expanders and superconcentrators.

The design of interconnection networks employs design-theoretic tools in a number of ways. The use of designs to cover all pairs of nodes is prevalent in diameter one networks; on the other hand, the balanced intersection of blocks is shown to lead to high expansion factors, and hence to highly connecting networks.

11. Distributed Consensus

In this section, we discuss a network topology and message-passing protocol proposed by Lakshman and Agrawala (1986) which allows for efficient computation of certain associative functions in a distributed network. The scheme is based on the use of finite projective planes.

Consider a database distributed over n sites (e.g. computers) which are to be linked in a decentralized communication network. A function which depends on the contents of the database is to be computed, and the value is to be made known to all sites. We assume that the evaluation is to be symmetric, and each site acts identically to all others; there is no site hierarchy. A naive method to compute the function is to have each site send its own data to all other sites; then each site can carry out the entire computation. This method involves $O(n^2)$ messages, and a single round of message passing.

The situation can be improved to a total cost of $O(n\sqrt{n})$ messages over two rounds of message passing, with each site sending $O(\sqrt{n})$ messages per round, as follows. Assume first that $n = m^2 + m + 1$ and that a projective plane on n points exists; we discuss how to modify the scheme to handle general n below. Let the sites correspond to the points of the plane. The *lines* (blocks) of the plane can be put into one-to-one correspondence with the points, such that each point is associated with a unique block containing it. (This "system of distinct representatives" can be found using a bipartite graph matching algorithm.) The point i associated with each block B_i is the *leader* of block B_i . Each point i sends messages to all other points in block B_i and to the leaders of all other blocks in which point i occurs. Then at each round, each point i receives exactly $2m$ messages, one from each of the $2m$ (distinct) points it sends messages to. Relying on the fact that the index in the corresponding design is $\lambda=1$, the communication paths outlined above provide a "backbone" in this topology such that two successive rounds of communication suffice for the propagation of data from each site to every other.

As an application, consider the use of this set-up and these communication paths to solve the *decentralized extrema finding* problem. Each of the n sites contains a value; the goal is to determine the maximum of the values, and to make this maximum known to all sites. Let $v(i)$ be the value at site i , and let each site i in the network carry out the following steps:

1. send $v(i)$ to the sites on its communication paths
2. await receipt of $2m$ values (messages)
3. compute the maximum M_i of these received values and $v(i)$
4. send M_i to the sites on its communication paths
5. await receipt of $2m$ values
6. compute the maximum M of these values and M_i .

It is easily proven that the value M computed by each site in this last step is the desired maximum. To verify the protocol on the projective plane of order 2, the reader may wish to simply use $v(i) = i$. We list the blocks here for convenience; block leaders are distinguished in boldface.

1 2 4	2 3 5	3 4 6	4 5 0	5 6 1	6 0 2	0 1 3
-------	-------	-------	-------	-------	-------	-------

As mentioned earlier, if $n = m^2 + m + 1$ for some $m = p^k$, then a projective plane on n points exists and can be used. Otherwise, a projective plane with point set of smallest cardinality exceeding n can be employed; "virtual sites" are then used in addition to the original n sites, and the communication scheme can be modified appropriately, affecting its complexity by only a constant factor. For consensus problems using symmetric protocols with no site hierarchy, as in the model discussed above, it is easily shown that $O(n\sqrt{n})$ is a lower bound on the number of messages required

to reach consensus. It follows that the protocol discussed above is optimal. We note that similar results can be obtained by more direct methods.

Decentralized extrema finding is of use, for example, in the coordination of distributed checkpoints. Other applications of this communication protocol include the computation of associative functions which have inverses, and commit protocols in distributed database systems.

12. Keys in Relational Databases

In this section, we consider another selection problem which has been solved using designs. A *relational database* can be viewed as an $m \times n$ matrix, with rows indexed by individuals I and columns indexed by attributes \mathcal{A} of the individuals. Suppose that the values of the attributes $A \subseteq \mathcal{A}$ are known for an individual; it may then happen that the value for an attribute $b \notin A$ is determined by the known values. This would happen exactly when all individuals having the same values for attributes in A also have the same value for the attribute b . If this occurs for all selections of values of the attributes in A , we say b is in the *closure* of A , $cl(A)$.

Consider a set A' with $cl(A') = \mathcal{A}$. In other words, all information about an individual can be deduced from its values for attributes in A' . A' is a *key* of the database. When A' is minimal, in that no subset $A'' \subset A'$ has $cl(A'') = \mathcal{A}$, A' is termed a *minimal key* of the database. Minimal keys are the simplest sets of attributes which are sufficient to recover all information about an individual. Hence the structure of minimal keys in a relational database holds much interest.

Armstrong (1974) and Békéssy and Demetrovics (1979) establish that any set of subsets

$$\{A_1, \dots, A_s\}, \quad A_i \subset \mathcal{A}, \text{ and } A_i \not\subseteq A_j \text{ for } i \neq j$$

can be realized as the set of minimal keys in some relational database. In fact, they show that a relational database can realize any *closure operation* defined by the following three axioms:

$$(A1) \quad A \subseteq cl(A)$$

$$(A2) \quad A \subseteq B \Rightarrow cl(A) \subseteq cl(B), \text{ and}$$

$$(A3) \quad cl(cl(A)) = cl(A).$$

Naturally, many different databases (matrices) realize the same closure operation; the minimum such matrix is the one with fewest individuals (rows).

Demetrovics, Füredi and Katona (1985) consider a specialized question of this form. Let \mathcal{L}_k^n be a closure operation on \mathcal{A} , $|\mathcal{A}| = n$, defined by

$$\mathcal{L}_k^n(A) = \begin{cases} A & \text{if } |A| < k \\ \mathcal{A} & \text{if } |A| \geq k \end{cases}$$

This situation is somewhat analogous to threshold schemes; here, every k -subset is a minimal key, but knowing fewer than k attribute values does not allow us to determine the value of any further attribute. Unlike perfect threshold schemes, however, knowing $k-1$ attribute values does enable us to eliminate many of the individuals from consideration.

From our earlier remarks, we know that the closure operation \mathcal{L}_k^n is realized by some assignment of attribute values to some set of individuals (i.e., by some matrix). How few individuals can realize the closure operation \mathcal{L}_k^n ? Let $s(\mathcal{L}_k^n)$ denote this minimum. Demetrovics, Füredi and Katona (1985) determine $s(\mathcal{L}_k^n)$ exactly for $k = 1, 2, n-1$, and n ; they obtain bounds for other values of k . For $k = 3$, they obtain quite tight bounds using designs. First, notice that for every 2-subset of attributes, there must be at least two individuals who agree in values of these attributes (otherwise the 2-subset is a key). No two individuals agree in values of three attributes (every 3-subset is a key). Hence the number of pairs of individuals must be at least the number of pairs of attributes; in other words, $s(\mathcal{L}_3^n) \geq n$.

To realize the minimum in this bound, every pair of individuals must agree in precisely two attributes. This minimum *can* be realized in certain cases using block designs. Let (V, \mathcal{B}) be a $(v, 3, 2)$ block design having a near-resolution into parallel classes P_1, \dots, P_v , so that P_i contains $\frac{v-1}{3}$ blocks whose union is $V \setminus \{v_i\}$. In addition, we require that for each i, j with $i \neq j$, there is exactly one pair appearing in a block of P_i and a block of P_j . Now construct a $v \times v$ matrix with diagonal entries all zero, and off-diagonal entry (i, j) equal to l if and only if v_i appears in the l^{th} triple of P_j . This matrix realizes the closure operation \mathcal{L}_3^v .

Demetrovics, Füredi and Katona (1985) construct the required $(v, 3, 2)$ designs for all $v \equiv 1, 4 \pmod{12}$ by replacing each block of a $(v, 4, 1)$ design by a $(4, 3, 2)$ design on the same points. In fact, Bennett and Zhu (1987) extend their results by using pairwise balanced designs. As a consequence, the bound $n \leq s(\mathcal{L}_3^n) \leq n+2$ holds for all $n \geq 20$.

The determination of $s(\mathcal{L}_3^n)$ is motivated initially by a problem in the theory of relational databases, but the design-theoretic aspect arises primarily in a specialized subproblem of the database problem; the main feature of this subproblem is to balance the appearance of subsets.

13. Summary

We have examined ten application areas of combinatorial designs in computer science. In each case, designs and related combinatorial configurations arise naturally. It is clearly unwise to try to characterize precisely those applications in which designs might prove useful; nevertheless, it is important to identify general paradigms for the application of designs. We make a first attempt at identifying paradigms here.

Designs provide balanced set systems. In most of the applications studied, a balance property is required. Balance on appearance of subsets arises in the design of core access switches to ensure zero-noise, in the design of threshold schemes to ensure that no partial information can be extracted, and in the design of authentication codes to achieve the desired secrecy properties and to minimize the probability of undetected deception. Balance on block sizes arises in combinatorial filing schemes to ensure that no bucket is too large, and in interconnection networks to ensure that no bus carries too much traffic. Balance on intersections arises in the design of expanding networks, and algorithms for sorting in rounds. We have also seen that balance has some algorithmic consequences; it is used to ensure good deterministic upper bounds on algorithms; remarkably, in other contexts, balance is used to force an algorithm to take exponentially many steps. There are two primary themes to the use of balance. Firstly, balance in designs leads to balanced load, when the blocks correspond to some physical entity (e.g., a bus or a 'bucket'). Secondly, balance ensures that limited partial information is obtained by examining a small set of elements. It is interesting to remark that the balance properties of designs are those most exploited in experimental design theory (see Street and Street (1987)).

Designs are minimum coverings, and maximum packings. We have seen covering and packing applications throughout this survey. Coverings arise from the need to represent all subsets of specified cardinality; we have seen this in combinatorial filing schemes, closure operations for relational databases, and interconnection networks. The minimum number of blocks in a covering is realized by a design (when the required design exists); hence, requirements for efficiency dictate the use of designs. Packings and set packings arise from the need to avoid redundant coverage. In our applications, packings arise in the design of core access switches, to limit the amount of noise generated. It is important to note here that the 'packing' aspect of designs is that most exploited in the design of error-correcting codes (see MacWilliams and Sloane (1978)).

While these two main themes capture the flavour (if not the details) of the applications discussed here, there are many less general themes which are still useful. We mention one here, because we believe that its importance has been generally underestimated. We have seen applications of resolutions and partitions of designs, particularly used as an additional balance property. However, resolutions of designs also arise naturally in scheduling problems. The reason for this is quite simple: a parallel class is

a partition of the elements into blocks. If each block corresponds to some task (or statistical test, or game, for example), all tasks in a parallel class can be performed concurrently. Moreover, completing all tasks corresponding to blocks of a resolvable design takes an amount of time equal to the number of parallel classes; this is clearly the minimum amount of time required to complete all tasks. Hence resolvable designs have been used widely in scheduling games (see Schreuder (1980), for example). Applications to timetabling have also been studied (see Hilton (1980, 1981)). Design-theoretic notions are also used in establishing the complexity of some scheduling problems (Colbourn (1983,1984)). However, applications to scheduling problems in computer science seem not to have been explored widely. Some first steps in this direction are taken in Bräsel (1988); we expect that resolvable designs will find reasonably wide application in scheduling problems.

We close with some words of warning. This is a first survey on applications of designs in computer science. We have been somewhat conservative in deciding what to include and what to omit. In general, we have omitted the vast amount of literature in which error-correcting codes find applications in computer science, despite the fact that design theory and coding theory are very closely linked. On the other hand, we have been more liberal in exploring applications of geometries in computer science, since we employ only design-theoretic aspects of the geometries. Despite the self-imposed restrictions to consider only applications in computer science, and only applications of designs, we have found a rich body of knowledge. Hence we conclude that designs are indeed useful tools in solving problems of computation effectively.

Acknowledgements

We are indebted to many people for assisting in locating the literature on design applications: Frank Bennett, Jean-Claude Bermond, Fan Chung, Karen Colbourn, Janelle Harms, Marlene Jones, Sanpei Kageyama, Richard Karp, Don Kreher, Rudi Mathon, Vojta Rödl, Alex Rosa, Doug Stinson, Victor Wei and Scott Vanstone. We also appreciate the efforts of many colleagues in making comments on the presentation, especially Rob Day, Dieter Jungnickel, Don Kreher, Bill Pulleyblank, Alex Rosa and Scott Vanstone. Research of the first author is supported by NSERC Canada under grant A0579. In addition, this research was supported in part by the Institute for Mathematics and Its Applications with funds provided by the National Science Foundation.

References

ABRAHAM, C.T., GHOSH, S.P. AND RAY CHAUDHURI, D.K. 1968. File organization schemes based on finite geometries, *Information and Control* 12 (2) 143-163.

- ALON, N. 1985. Expanders, sorting in rounds and superconcentrators of limited depth, *Proc. Seventeenth ACM Symposium on the Theory of Computing*, pp. 98-102.
- ARMSTRONG, W.W. 1974. Dependency structures of database relationship, *Proceedings of IFIP74*, North-Holland, pp. 580-583.
- AVIS, D. 1980. A note on some computationally difficult set covering problems, *Math. Programming* 18 (2) 138-145.
- AZAR, Y. AND VISHKIN, U. 1987. Tight comparison bounds on the complexity of parallel sorting, *SIAM Journal on Computing* 16 (3) 458-464.
- BABAI, L. AND ERDOS, P. 1982. Representation of group elements as short products, *Annals Discrete Math.* 15, 27-30.
- BABAI, L., FRANKL, P. AND SIMON, J. 1986. Complexity classes in communication complexity theory, *Proc. Twenty-Seventh Conf. Foundations of Computer Science*, pp. 337-347.
- BEKESSY, A. AND DEMETROVICS, J. 1979. Contribution to the theory of database relations, *Discrete Mathematics* 27 (1) 1-10.
- BENNETT, F.E. AND ZHU, L. 1987. private communications.
- BERLEKAMP, E.R. 1970. Factoring polynomials over large finite fields, *Mathematics of Computation* 24 (111) 713-735.
- BERMAN, G. 1976. The application of difference sets to the design of a balanced multiple-valued filing scheme, *Information and Control* 32 (2) 128-138.
- BERMOND, J.C. AND BOND, J. 1986. Combinatorial designs and hypergraphs of diameter one, *LRI Rapport de Recherche No. 329*, Centre d'Orsay, Université de Paris-Sud, December.
- BERMOND, J.C., BOND, J. and SACLE, J.F. 1984. Large hypergraphs of diameter 1, *Graph Theory and Combinatorics, Proc. Coll. Cambridge, 1983*, pp. 19-28.
- BETH, T., JUNGnickEL, D. AND LENZ, H. 1985. *Design Theory*, Bibliographisches Institut, Mannheim.
- BEUTELSPACHER, A. 1987. Geometric structures as threshold schemes, preprint.
- BEUTELSPACHER, A., JUNGnickEL, D., VAN OORSCHOT, P.C. AND VAN-
STONE, S.A. 1987. Pair-splitting sets in $AG(m,q)$, Research Report CORR 87-41 (December 1987), Faculty of Mathematics, University of Waterloo.
- BLACHMAN, N.M. 1956. On the wiring of two-dimensional multiple-coincidence magnetic memories, *IEEE Transactions on Electronic Computers* EC-5 (1) 19-21.
- BLAKLEY, G.R. 1979. Safeguarding cryptographic keys, *AFIPS 1979 National Computer Conference* Volume 48, 313-317.

- BOLLOBAS, B. AND ROSENFELD, M. 1981. Sorting in one round, *Israel Journal of Mathematics*, 38 (1-2) 154-160.
- BOSE, R.C., ABRAHAM, C.T. AND GHOSH, S.P. 1969. File organization of records with multiple-valued attributes for multi-attribute queries, in: *Combinatorial Mathematics and Its Applications* (R.C. Bose and T.A. Dowling, editors) UNC Press, Chapel Hill, pp. 277-297.
- BOSE, R.C. AND KOCH, G.G. 1969. The design of combinatorial information retrieval systems for files with multiple-valued attributes, *SIAM Journal on Applied Mathematics* 17 (6) 1203-1214.
- BRÄSEL, H. 1988. Rangminimale Maschinenbelegungsprobleme, preprint.
- BRICKELL, E.F. 1984. A few results in message authentication, *Congressus Numerantium* 43, 141-154.
- BROUWER, A.E. 1976. On associative block designs, in: *Combinatorics* (A. Hajnal and V.T. Sós, editors), North-Holland, pp. 173-184.
- BUCHOLZ, W. 1963. File organization and addressing, *IBM Systems Journal* 2 (June) 86-111.
- BURKHARD, W.A. 1976a. Partial match retrieval, *BIT* 16 (1) 13-31.
- BURKHARD, W.A. 1976b. Hashing and trie algorithms for partial match retrieval, *ACM Trans. Database Systems* 1 (2) 175-187.
- CAMION, P. 1983. A deterministic algorithm for factorizing polynomials of $F_q[x]$, *Annals Discrete Math.* 17, 149-157.
- CHIEN, R.B. 1959. Orthogonal matrices, error-correcting codes, and the design of efficient load-sharing matrix switches, *IEEE Transactions on Electronic Computers* EC-8 (3) 400.
- CHIEN, R.B. 1960. A class of optimal noiseless load-sharing core switches, *IBM Journal of Research and Development* 4 (4) 414-417.
- CHOR, B. AND GOLDBREICH, O. 1985. Unbiased bits from sources of weak randomness and probabilistic communication complexity, *Proc. Twenty-Sixth Conf. Foundations of Computer Science*, pp. 429-442.
- CHOW, D.K. 1969. New balanced file-organization schemes, *Information and Control* 15 (5) 377-396.
- CHUNG, F.R.K. 1979. On concentrators, superconcentrators, generalizers and non-blocking networks, *Bell Sys. Tech. J.* 58 (8) 1765-1777.
- CHUNG, F.R.K., SALEHI, J.A. AND WEI, V.K. 1987. Optical orthogonal codes: design, analysis and applications, preprint.
- COLBOURN, C.J. 1983. Embedding partial Steiner triple systems is NP-complete, *J. Combin. Theory* A35 (1) 100-105.

- COLBOURN, C.J. 1984. The complexity of completing partial Latin squares, *Discrete Applied Mathematics* 8 (1) 25-30.
- COLBOURN, M.J. 1985. Algorithmic aspects of combinatorial designs: a survey, *Annals Discrete Math.* 26, 67-136.
- COLBOURN, M.J. AND MATHON, R.A. 1980. On cyclic Steiner 2-designs, *Annals Discrete Math.* 7, 215-253.
- CONSTANTINE, G. Jr. 1958. A load-sharing matrix switch, *IBM Journal of Research and Development* 2 (3) 204-211.
- CONSTANTINE, G. Jr. 1960. New developments in load-sharing matrix switches, *IBM Journal of Research and Development* 4 (4) 418-422.
- DE BRANDES, M. AND RÖDL, V. 1984. Steiner triple systems with small maximal independent sets, *Ars Combinatoria* 17 (June) 15-19.
- DEMETROVICS, J., FUREDI, Z. AND KATONA, G.O.H. 1985. Minimal matrix representations of closure operations, *Discrete Applied Mathematics* 11 (2) 115-128.
- DOYEN, J. AND ROSA, A. 1980. An updated bibliography and survey of Steiner systems, *Annals Discrete Math.* 7, 317-349.
- ERDŐS, P. AND LARSON, J. 1982. On pairwise balanced block designs with the sizes of blocks as uniform as possible, *Annals Discrete Math.* 15, 129-134.
- FRANKL, P., RÖDL, V. AND WILSON, R.M. 1987. The number of submatrices of given type in a Hadamard matrix and related results, *J. Combin. Theory B*, to appear.
- FULKERSON, D.R., NEMHAUSER, G. AND TROTTER, L. 1974. Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems, *Math. Programming Study* 2, 72-81.
- GHOSH, S.P. AND ABRAHAM, C.T. 1968. Application of finite geometry in file organization for records with multiple-valued attributes, *IBM Journal of Research and Development* 12 (2) 180-187.
- HAGGKVIST, R. AND HELL, P. 1981. Parallel sorting with constant time for comparisons, *SIAM Journal on Computing*, 10 (3) 465-472.
- HAGGKVIST, R. AND HELL, P. 1982. Sorting and merging in rounds, *SIAM Journal on Discrete and Algebraic Methods*, 3 (4) 465-473.
- HILTON, A.J.W. 1980. The reconstruction of latin squares with applications to school timetabling and to experimental design, *Mathematical Programming Study* 13, 68-77.
- HILTON, A.J.W. 1981. School timetables, *Studies on Graphs and Discrete Programming*, (P. Hansen, ed.), North-Holland Publishing Company, pp. 177-188.

- HUGHES, D.R. AND PIPER, F.C. 1985. *Design Theory*, Cambridge University Press, Cambridge, UK.
- KARP, R.M., UPFAL, E. AND WIGDERSON, A. 1985. The complexity of parallel computations on matroids. *Proc. Twenty-Sixth Conf. Foundations of Computer Science*, pp. 541-550.
- KARP, R.M. AND WIGDERSON, A. 1985. A fast parallel algorithm for the maximal independent set problem, *Journal of the ACM* 32 (4) 762-773.
- KILIAN, J., KIPNIS, S. AND LEISERSON, C.E. 1987. The organization of permutation architectures with bussed interconnections, *28th Symposium on Foundations of Computer Science*, pp. 305-315.
- KOCH, G.G. 1969. A class of covers for finite projective geometries which are related to the design of combinatorial filing schemes, *Journal of Combinatorial Theory* 7 (3) 215-220.
- LAKSHMAN, T.V. AND AGRAWALA, A.K. 1986. Efficient decentralized consensus protocols, *IEEE Transactions on Software Engineering* SE-12 (5) 600-607.
- MACWILLIAMS, F.J. AND SLOANE, N.J.A. 1978. *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam.
- MARCUS, M.P. 1959. Doubling the efficiency of the load-sharing matrix switch, *IBM Journal of Research and Development* 3 (2) 195-196.
- MATHON, R. AND ROSA, A. 1985. Tables of parameters of BIBDs with $r \leq 41$ including existence, enumeration, and resolvability results, *Annals Discrete Math.* 26, 275-308.
- MICKUNAS, M.D. 1980. Using projective geometry to design bus connection networks, *Proceedings of the Workshop on Interconnection Networks for Parallel and Distributed Processing*, April, pp. 47-55.
- MINNICK, R.C. AND HAYNES, J.L. 1962. Magnetic core access switches, *IEEE Transactions on Electronic Computers* EC-11 (3) 352-368.
- ODLYZKO, A. 1985. Discrete logarithms in finite fields and their cryptographic significance, *Advances in Cryptology (Proceedings of EUROCRYPT '84)*, (T. Beth, N. Cot, I. Ingemarsson, ed.), Springer-Verlag, pp. 224-314.
- PIPPENGER, N. 1987. Sorting and selecting in rounds, *SIAM Journal on Computing* 16 (6) 1032-1038.
- RABIN, M.O. 1980. Probabilistic algorithms for finite fields, *SIAM Journal on Computing* 9 (2) 273-280.
- RAGHAVARAO, D. 1971. *Constructions and Combinatorial Problems in Design of Experiments*, Wiley, New York.

- RAY-CHAUDHURI, D.K. 1968. Combinatorial information retrieval systems for files, *SIAM Journal on Applied Mathematics* 16 (5) 973-992.
- RIVEST, R.L. 1974a. On hash-coding algorithms for partial-match retrieval, *Proc. Fifteenth ACM Symposium on the Theory of Computing*, pp. 95-103.
- RIVEST, R.L. 1974b. On the optimality of Elias's algorithm for performing best-match searches, *Proc. IFIP74*, pp. 678-681.
- RIVEST, R.L. 1976. Partial-match retrieval algorithms, *SIAM Journal on Computing* 5 (1) 19-50.
- RODL, V. 1985. On a packing and covering problem, *Eur. J. Combinatorics* 6 (1) 69-78.
- SCHREIBER, S. 1973. Covering all triples on n marks by disjoint Steiner systems, *J. Combin. Theory A* 15, 347-350.
- SCHREUDER, J.A.M. 1980. Constructing timetables for sport competitions, *Mathematical Programming Study* 13, 58-67.
- SHAMIR, A. 1979. How to share a secret, *Communications of the ACM* 22 (11) 612-613.
- SIMMONS, G.J. 1984. Message authentication: a game on hypergraphs, *Congressus Numerantium* 45, 161-192.
- SINGLETON, R.C. 1962. Load-sharing core switches based on block designs, *IEEE Transactions on Electronic Computers* EC-11 (3) 346-352.
- STINSON, D.R. 1987. A construction for authentication/secrecy codes from certain combinatorial designs, preprint.
- STINSON, D.R. 1988. Some constructions and bounds for authentication codes, *Journal of Cryptology*, to appear.
- STINSON, D.R. AND VANSTONE, S.A. 1988. A combinatorial approach to threshold schemes, to appear.
- STREET, A.P. AND STREET, D.J. 1987. *Combinatorics of Experimental Design*, Clarendon Press, Oxford.
- TAKAHASHI, I. 1973. Combinatorial filing schemes (in Japanese), *Bulletin of the Institute for Research in Productivity (Waseda University)* 4, 49-74.
- TEIRLINCK, L. 1987. Non-trivial t -designs without repeated blocks exist for all t , *Discrete Mathematics* 65 (3) 301-311.
- VAN OORSCHOT, P.C. AND VANSTONE, S.A. 1987. On splitting sets in block designs and finding roots of polynomials, preprint.
- VAN SLYKE, R.M. 1982. Redundant set covering in telecommunication networks, *Proc. 1982 IEEE Large Scale Systems Symposium*, October 1982, pp. 217-222.

- WILSON, R.M. 1972a. An existence theory for pairwise balanced designs I. Composition theorems and morphisms, *J. Combin. Theory* A13 (2) 220-245.
- WILSON, R.M. 1972b. An existence theory for pairwise balanced designs II. The structure of PBD-closed sets and the existence conjectures, *J. Combin. Theory* A13 (2) 246-273.
- WILSON, R.M. 1974. Some partitions of all triples into Steiner triple systems, *Lecture Notes in Mathematics* 411, pp. 267-277.
- WILSON, R.M. 1975. An existence theory for pairwise balanced designs III. Proof of the existence conjectures, *J. Combin. Theory* A18 (1) 71-79.
- ZAITSSEV, G.V., ZINOVIEV, V.A. AND SEMAKOV, N.V. 1973. Interrelation of Preparata and Hamming codes and extension of Hamming codes to new double-error-correcting codes, *Proc. Second International Symposium on Information Theory, Tsahkadsor, USSR, Akademia Kiado, Budapest*, pp. 257-263.
- ZIERLER, N. 1974. A conversion algorithm for logarithms in $GF(2^n)$, *Journal of Pure and Applied Algebra* 4, 353-356.

Appendix: An Overview of Design Theory

We first review some basic conditions on existence of non-trivial block designs and t -designs. We then examine some combinatorial configurations which relax the strong requirements for designs.

A.1 Necessary conditions for existence

Let us suppose that a t -(v, k, λ) design (V, \mathcal{B}) exists. Consider a subset $X \subseteq V$ with $0 \leq |X| < t$. The total number of t -subsets of V containing X in \mathcal{B} is $\lambda \binom{v-|X|}{t-|X|}$, while any block containing X contains $\binom{k-|X|}{t-|X|}$ of these t -sets. Hence by considering all possible sizes of X , we obtain t divisibility conditions:

$$\binom{k-i}{t-i} \mid \lambda \binom{v-i}{t-i} \quad \text{for } i=0, \dots, t-1.$$

For example, for a 3-($v, 4, 1$) design to exist, we must have

$$4 \mid \binom{v}{3}, \quad 3 \mid \binom{v-1}{2}, \quad \text{and} \quad 2 \mid v-2,$$

from which we obtain the "congruence condition" $v \equiv 2 \text{ or } 4 \pmod{6}$. Suppose a subset X is chosen, and let the subset of blocks in \mathcal{B} each containing the set X be \mathcal{B}_X . It is then easy to check that $(V \setminus X, \mathcal{B}_X \setminus X)$ is a $(t-|X|)$ -design, where $\mathcal{B}_X \setminus X$ is the set of blocks obtained by deleting from each block in \mathcal{B}_X all points in X . This is called the *derived* design for X .

For block designs ($t = 2$), we obtain the simpler necessary conditions

$$r(k-1) = \lambda(v-1) \quad \text{and} \quad rv = bk.$$

These can be obtained by counting, in two different ways each, the number of pairs that a given point appears in, and the number of points in the entire design, respectively.

A second type of necessary condition arises by considering the number of blocks. The number of blocks b in a block design is equal to $\lambda \binom{v}{2} / \binom{k}{2}$; a well-known inequality, Fisher's inequality, shows that $b \geq v$ in any block design, and hence we have $\lambda(v-1) \geq k(k-1)$.

A.2 Some known sufficient conditions

Existence problems for block designs and t -designs are far from settled in general. We only summarize some main existence results here. For block designs, an elegant theory due to Wilson (1972a, 1972b, 1975) establishes that the necessary conditions for

the existence of a (v, k, λ) design are sufficient for v sufficiently large with respect to k . Hence existence of block designs is, in an asymptotic sense, well understood; nevertheless, complete solutions are known only for (v, k, λ) designs with $k = 3, 4$, and 5 .

For t -designs with $t > 2$, much less is known. Teirlinck (1987) recently proved that simple t -designs exist for all values of t . However, except for $3-(v, 4, \lambda)$ designs, the necessary conditions are not known to be sufficient, even in an asymptotic sense. In fact, t -designs with index $\lambda = 1$ (called *Steiner systems*) are at present unknown for $t > 5$. Much of the effort in combinatorial design theory has been invested in constructing designs with additional properties. Most effort to date in establishing existence results has been invested in triple systems $((v, 3, \lambda)$ designs), quadruple systems $(3-(v, 4, \lambda)$ designs), and Steiner systems $(t-(v, k, 1)$ designs). We do not attempt to review this literature here; see Doyen and Rosa (1980) for a bibliography, and Mathon and Rosa (1985) for a summary of existence results for "small" parameter sets.

A.3 Symmetric Designs

In many of the problems which we discuss, an important aspect of the block designs used is the number of blocks; in many applications, this number must be minimized. We have seen that Fisher's inequality requires $b \geq v$; the minimum number of blocks is realized when equality holds. Such a design is called a *symmetric* design. In a symmetric design, we have $\lambda(v-1) = k(k-1)$, and hence the parameters of a symmetric design are of the form $(\frac{k^2-k}{\lambda}+1, k, \lambda)$; note that $b = v$ implies $k = r$. The order of a symmetric design is $n = k - \lambda$.

The case $\lambda=1$ has received special attention. A symmetric design with parameters $(k^2-k+1, k, 1)$ is called a *(finite) projective plane*; the parameters can equivalently be written as $(n^2+n+1, n+1, 1)$, and the plane is then of order n . From a symmetric design (V, \mathcal{B}) , one can form a *residual* design by selecting one block, removing that block and removing all of its elements from the remaining blocks. Residuals of projective planes are $(n^2, n, 1)$ designs, usually called *affine planes*.

Projective planes can be obtained from a general class of structures which give rise to symmetric designs. Let S be an $(m+1)$ -dimensional vector space over $GF(q)$, the finite field with q elements, where q is a prime or prime power. The set of all subspaces of S is called the *projective geometry of dimension m over $GF(q)$* , denoted $PG(m, q)$. The 1-dimensional and m -dimensional subspaces of S are called *points* and *hyperplanes*, respectively. For each hyperplane H , let B_H be the set of points contained in H . Then using the 1-dimensional subspaces of S as points, the block set $\{B_H: H \subset S\}$ defines a symmetric design with parameters

$$v = \frac{q^{m+1}-1}{q-1}, \quad k = \frac{q^m-1}{q-1}, \quad \lambda = \frac{q^{m-1}-1}{q-1}.$$

Taking $m=2$ yields projective planes. It follows that projective planes are known to exist for all values $n=q$ which are powers of primes; at present no planes of non-prime power order are known.

In a nontrivial symmetric design of order n , $4n-1 \leq v \leq n^2+n+1$. Projective planes realize the maximum here. At the other extreme, the symmetric designs with parameters $(4n-1, 2n-1, n-1)$ are *Hadamard designs*, and arise from related configurations called Hadamard matrices. Existence of Hadamard designs is still unsettled, but numerous infinite families of such designs are known. (The smallest open case is $n = 107$.)

We have already mentioned one reason for special interest in symmetric designs: they minimize the number of blocks in a design. A second reason is equally important. Two blocks in a symmetric (v, k, λ) design always intersect in precisely λ elements. The reason for this, and the generalization to designs in general, is our next topic.

A.4 Duals of designs and set packings

A *set packing* with intersection number λ is a set system (W, \mathcal{C}) in which any two different sets in \mathcal{C} intersect in at most λ elements. It is an *exact* set packing if any two different sets in \mathcal{C} intersect in precisely λ elements. It is *uniform* precisely when the set system is k -uniform for some k . Exact set packings balance the intersection of blocks rather than the appearance of subsets; however, this notion is closely related to balance in designs.

From a block design (V, \mathcal{B}) , we can form a *dual* as follows. Form a set W whose elements are in one-to-one correspondence with the sets of \mathcal{B} . Now for each $v \in V$, form a set $C_v \subseteq W$ consisting of all elements corresponding to blocks to which v belongs. Now let $\mathcal{C} = \{C_v : v \in V\}$. (W, \mathcal{C}) is again a set system. If (V, \mathcal{B}) has replication number r , (W, \mathcal{C}) is r -uniform. Let λ be the index of \mathcal{B} . Consider two sets in \mathcal{C} ; they must intersect in precisely λ elements. (W, \mathcal{C}) is therefore a uniform exact set packing with intersection number λ . In general, a uniform exact set packing need not itself be a design. However, the dual of a symmetric design is also a (symmetric) design; hence we see that symmetric designs are precisely uniform exact set packings with a *maximum* number of sets. It is worth mentioning at this point that set packings are closely related to error-correcting codes; the intersection property guarantees that no two sets share many elements, corresponding to a guaranteed minimum distance in the related code.

A uniform set packing (not necessarily exact now) with intersection number λ requires only the less stringent restriction that sets intersect in at most λ elements; again we have that the maximum number of sets is achieved by the dual of a design (when the required design exists). Of course, we need not require that a set packing

(or exact set packing) be uniform; in general, we permit different block sizes.

A.5 Packings and Coverings

In view of our observations that much remains to be settled concerning existence of designs, and that the necessary conditions rule out many orders, it is reasonable to try to "come close". Hence we might relax some of the restrictions. Suppose that (V, \mathcal{B}) is a set system which is k -uniform on v elements and each t -subset appears *at most (at least)* λ times in blocks of \mathcal{B} ; then we call (V, \mathcal{B}) a t -(v, k, λ) *packing* (t -(v, k, λ) *covering*, respectively).

A t -(v, k, λ) packing can have at most $b(t, v, k, \lambda) = \lambda \binom{v}{t} / \binom{k}{t}$ blocks, while a t -(v, k, λ) covering must have at least this number. Equality holds if and only if the packing (covering) is a t -(v, k, λ) design. However, Rödl (1985) demonstrates that as v goes to infinity, the size of a maximum t -(v, k, λ) packing is $(1 - o(1))b(t, v, k, \lambda)$, and hence that we can always come "close" to a design.

Packings are also often called *partial* designs.

A.6 Pairwise balanced designs

Packings and coverings relax the requirement that the index be constant; here we relax instead the requirement that the block size be constant. A *pairwise balanced design (PBD)* with parameters (v, K, λ) is a set system on v elements with block sizes from K , and which is 2-balanced with index λ . A PBD does not in general have a unique replication number. Block designs are just PBDs with a single block size. In addition, PBDs are precisely the duals of (not necessarily uniform) exact set packings.

While there is a rich theory of pairwise balanced designs, we only remark on a few facts which we employ. First, Fisher's inequality applies to PBDs; hence symmetric designs are again PBDs with the minimum number of blocks. Second, Wilson's asymptotic existence theory applies to PBDs as well, and hence existence of a desired PBD is assured for v sufficiently large, provided that basic numerical conditions are met. Finally, one can both relax restrictions on block sizes and impose only an upper bound on the index; the result is a partial PBD (equivalently, a non-uniform packing).

A.7 Resolutions of designs

One property of designs which arises in numerous design applications deserves special attention. For a design (V, \mathcal{B}) , a *parallel class* (or *resolution class*) of blocks $\mathcal{P} \subseteq \mathcal{B}$ is a set of blocks such that no two intersect, and the union of all blocks of \mathcal{P} is V ; a *near-parallel class* is similar, but the union contains all but one element of V .

When \mathcal{B} can be partitioned into parallel classes, this partitioning is a *resolution*, and (V, \mathcal{B}) is a *resolvable* design. The necessary condition $v \equiv k \pmod{k(k-1)}$ is asymptotically sufficient for the existence of a resolvable $(v, k, 1)$ block design.

More generally, a $t-(v,k,\lambda)$ design may be *partitionable* into $(t')-(v,k,\lambda')$ designs; resolution is just the case $t'=1$ and $\lambda'=1$. We see applications for partitionable designs, but especially for the restricted case, resolvable designs.

A.8 Automorphisms of Designs

In the applications which we discuss, we often choose a design with some symmetry, or nontrivial automorphism. An *automorphism* of a design is a bijection from the elements onto themselves, which induces a bijection from the blocks into themselves. Provided that an automorphism is known, the design has a compact representation, in which a representative for each equivalence class (orbit) of blocks under the action of the automorphism is retained. The existence of such a compact representation enables one to find and use much larger designs in practical applications.

Of particular interest are designs with a *cyclic* automorphism, which is a cycle involving all elements of the design. Colbourn and Mathon (1980) provide a survey on cyclic designs, and remark on the importance of the compact representation here; a more recent example of the use of cyclic designs appears in Chung, Salehi and Wei (1987).