Balanced Incomplete Block Designs (BIBDs)

Keenan Hawekotte

May 2014

Abstract

We define and investigate Balanced Incomplete Block Designs (BIBDs), three theorems pertaining to BIBDs, and explore their usage in error correcting codes.

1 Introduction

Imagine, we're handed a piece of paper. On this paper is a list of sixteen dishwasher detergents, new and old, the company we work for is testing. We have been tasked with determining which of the products are able to be safely mixed. Due to limitations in the company, and our boss breathing down our necks, we can only mix up to four of the detergents at one time. Luckily, we are told that if any safety issues arise in our trials, it is caused by exactly two of the mixed detergents. Though, after handing us the paper, our boss says, "These tests better be done efficiently. The longer it takes, the longer the company loses money. Your job is on the line." With these added pressures, we must determine a series of tests that allow us to satiate our company's hunger for new and safe products.

An initial idea might be to take every possible combination of just two of the products and test them at one time. With our sixteen possible detergents, this leaves us with $\binom{16}{2} = 120$ trials that we need to run. We are certainly able

to run all 120 trials, but we are missing one key thing; we are not utilizing the full capacity of our mixtures. We can mix up to four detergents at a single time, but with this solution we are only mixing two. This might very well leave the boss enraged because we were not as efficient as possible.

In order to use the full capacity allotted for use, let us simply mix all possible combinations of four detergents. Note, we have to use all possible combinations of four detergents because if we omit any particular combination of four then there is uncertainty as to whether or not the mixture of detergents omitted would be safe for usage. This means that if we leave out some combination, and we were asked to prove that it was safe, we would have no actual proof and the company would be at risk. This again would leave our boss engulfed in anger. So we decide to use all possible combinations of four of the sixteen detergents. This means that we need to run $\binom{16}{4} = 1820$ trials. This number is simply not feasible. The time and man-hours needed to complete this task would not be worth our time nor the company's. Thus, this idea would leave us with a despondent boss and most likely out of a job.

A final idea would be to impose one more rule on our tests, that each pair of detergents is mixed together and tested in exactly two trials. With this new rule, if there are any irregularities in our tests we are able to trace the issue back to the detergent that caused the problem using simple logic. For example, if we are testing a mixture composed of detergents 1-4, and they are not safe for use, then we know from our assumption that each mixed detergent is only involved in one other trial with each other detergent. Thus, we can examine those other trials and we would expect to find that one of the other trials would show the same breach in safety caused by our initial test. Now after applying this new rule, we are able to use balanced incomplete block designs (BIBDs) to create an experiment that will meet the efficiency and time constraints imposed

by our boss. Using BIBDs, we end up discovering that we would only need 20 trials to test all possible combinations using all available space per trial. Thus, leaving us with a job, our boss ecstatic, and our company with money.

1.1 What is a Balanced Incomplete Block Design?

We begin by defining a balanced incomplete block design (BIBD).

Definition 1.1. *A **balanced incomplete block design (BIBD)** is a k-uniform, r-regular incomplete block design, with each pair of vertices occurring together exactly in exactly λ blocks. We say the BIBD is of parameters (b, v, r, k, λ) or that the design is a (b, v, r, k, λ)-design.*

Now that we have the full definition, we are able to pick it apart. At the heart of the BIBDs lies the idea of a block design.

Definition 1.2. *A **block design** is a set together with a collection of subsets called* blocks *(repetition sometimes allowed) that meet a specific set of criteria. The elements of the set are called* vertices*.*

Once we have established a block design, there are several other conditions that must be met before we have our wanted BIBD.

Next, a block design can be either be incomplete or complete.

Definition 1.3. *If a block design has a collection of subsets in which one subset does not contain all elements of the original set, it is **incomplete**. If a block design has a collection of subsets in which every subset contains all of the elements of the original set, it is **complete**.*

It is uncommon to find a complete block design due to the fact that all of the subsets are the original set itself. Complete designs are uninteresting themselves, unless further restrictions are placed on the subsets. In addition,

a BIBD requires an incomplete block design, and thus, we will only concern ourselves with incomplete block designs when discussing BIBDs.

Now, there are two more parts to BIBDs that need to be discussed, uniformity and regularity.

Definition 1.4. *If a design contains exactly k vertices in each block, then the design is said to be* ***uniform*** *or* ***k-uniform****.*

Definition 1.5. *If each vertex in a design occurs in exactly r blocks, then the design is said to be* ***regular*** *or* ***r-regular****.*

Both definitions above are vital to the definition of a BIBD. They ensure that the blocks within the BIBD are all of the same size, while ensuring that every vertex is in the same number of blocks. This allows for controlled testing environments and evenness throughout a BIBD. Extending the idea behind regularity into pairs of vertices, we discover the final component of a BIBD. If each pair of vertices occurs in exactly λ blocks, then we are able to construct a BIBD if all previous definitions are also met.

1.2 BIBD Parameters

To summarize from Section 1.1, we find that for a BIBD of parameters (b, v, r, k, λ),

- b is the total number of blocks in our design,
- v is the total number of vertices in our design,
- r is the number of blocks containing a point (refers to regularity),
- k is the number of vertices in each block (refers to uniformity),
- λ is the number of blocks containing each pair of vertices.

In the introduction, we were met with a challenge from our boss and determined that BIBDs would be the best way to solve our problem. So we must

identify what each of the parameters for the BIBD would be for our given situation.

We know that we have an initial set of 16 detergents. We also decided that we would use a mixture of 4 detergents from our list in each of the trials. So right away we have found some of our parameters. Since we have 16 total detergents, we have 16 vertices. Thus, $v = 16$. Considering each trial a block, we know that since there are 4 detergents in each trial, we have k-uniformity where $k = 4$. Now, we must remember the new rule we imposed, that each pair of detergents only be mixed together in exactly 2 trials. This is by definition λ, and therefore, $\lambda = 2$. Using this rule we are also able to logically determine our value for r. Since we have a total of 16 detergents, each separate detergent must be matched up with 15 others. Now, each trial has four spots, but one will be taken up by the detergent we want to be matched with the others. This leaves 3 open spots per trial, and thus we would need 5 trials per detergent to attain our wanted comparisons. Thus, we have $r = 5$ since each detergent must be in exactly 5 trials. Now, 5 trials for each of the 16 detergents implies that we need a total of 80 trials where each detergent is a part of exactly 5 of those trials. Because we are able to mix four detergents at a time, we find that we would need exactly 20 trials. Therefore, we have determined our number of blocks, and thus determined our number of trials, to be $b = 20$. We have now logically deduced all parameters of our wanted BIBD.

To summarize our parameters, we found,

- $b = 20$,
- $v = 16$,
- $r = 5$,
- $k = 4$,

- $\lambda = 2$.

The next step would be to determine what each trial looks like for our entire experiment using logical placements and matchings. This step will be omitted, but if more information is desired, please refer to chapter 17 of Bòna [1].

2 Selected Theorems

With BIBDs, comes a slew of useful theorems. We will discuss three of them in this section.

The first theorem provides us with a way to validate and ensure that we are able to create a BIBD based upon given criteria. In addition, part of the validation process involves determining the BIBD parameters, and thus the proof also acts as a framework for how to logically deduce these parameters.

Theorem 2.1. *For $1 < k < n$, the family of k-element subsets of $[n]$ is a BIBD with parameters, $(\binom{n}{k}, n, \binom{n-1}{k-1}, k, \binom{n-2}{k-2})$.*

Proof. To show that the family of k-element subsets of $[n]$ is a BIBD, we must be able to deduce values for each of the parameters. Thus, we know that since we are dealing with $[n]$, we only have n integers at our disposal. Therefore, we have a total of n vertices and $v = n$. Next, all k-element subsets of $[n]$ can be represented by $\binom{n}{k}$. Thus, there are a total of $\binom{n}{k}$ subsets (blocks), and we have now found that $b = \binom{n}{k}$. Since every subset will have k elements by assumption, we have uniformity with $k = k$. Now, to ensure regularity, we want each element of the subset to be matched with all $k-1$ other elements. There are $\binom{n-1}{k-1}$ ways of accomplishing this since there are now $n-1$ numbers (due to the fact that a number cannot be matched with itself) that we are able to choose our total of $k-1$ elements from. Thus, each element will have to be in $\binom{n-1}{k-1}$ blocks. Our last element we must determine is λ. Using the same logic as when we found

the value for regularity, we know that λ involves a pair of vertices. Thus, we now have two integers that cannot be matched with themselves and we know there are a total of $\binom{n-2}{k-2}$ ways that these two numbers can be paired with the other $k-2$ remaining integers. Therefore, $\lambda = \binom{n-2}{k-2}$.

We have now found all values necessary for the family of k-element subsets of $[n]$ to be a BIBD. This BIBD has parameters, $(\binom{n}{k}, n, \binom{n-1}{k-1}, k, \binom{n-2}{k-2})$. □

The use of the next theorem is similar to that of the first. It allows us to not only create BIBDs based off of partially given parameters, but it also allows us to ensure that a given parameter is in fact a BIBD.

Theorem 2.2. *For a BIBD of parameters (b, v, r, k, λ), $bk = vr$.*

Proof. Let S be a set with collection of blocks $\mathcal{B}$ that satisfies the conditions necessary to be a BIBD. We define the pair (w, B), where $w \in S$, $B \in \mathcal{B}$, and thus $w \in B$. Looking at the equation $bk = vr$, the right-hand side counts the number of pairs (w, B). This arises because we have v vertices, and we know that according to regularity, each vertex will be in r blocks. Thus, there are vr pairs necessary to describe all possible combinations of vertices given their regularity. Now, the left-hand side of the equation counts the exact same thing! This arises because in each block we have k elements by the definition of uniformity. So for each block there are k pairs for that block, and given b blocks, we have bk pairs necessary to describe all possible pairs given the number of blocks and how many elements are within them. □

Note that the theorem above is independent of λ, and thus is applicable not only to BIBDs, but also to any regular, uniform block design.

Now that we are sure $bk = vr$, we can show an example. In the problem introduced in the introduction, we found that we have values of $b = 20$, $v = 16$, $k = 4$, and $r = 5$. Thus, $bk = vr = (20)(4) = (16)(5) = 80$ and we are ensured

our introductory example is indeed a BIBD.

The next theorem serves the same purpose as Theorem 2.2, but is now only applicable to BIBDs since it includes the value λ.

Theorem 2.3. *For a BIBD of parameters* (b, v, r, k, λ), $r(k-1) = \lambda(v-1)$.

Proof. Let S be a set with collection of blocks $\mathcal{B}$ that satisfies the conditions necessary to be a BIBD. Now fix a vertex, x. We define the pair (w, B), where $w \in S$, $B \in \mathcal{B}$, $w \in B$, and $w \neq x$. Looking at the equation $r(k-1) = \lambda(v-1)$, we see that the left-hand side counts the number of pairs (w, B). This arises because we can choose any of the r blocks that contain x, and then we can choose any of the remaining $k-1$ vertices of B as w. Looking at the right-hand side, we see it counts the same thing! Though this side begins by choosing w in one of the $v-1$ possible ways (since $w \neq x$) and then chooses one of the λ remaining blocks that contain both x and w. □

An important note involving both Theorem 2.2 and Theorem 2.3, is that we only need the values of v, k, and λ, since the other values can be determined using the theorems. Thus, it is also common to see BIBDs written as v, k, λ-designs.

3 Error Correcting Codes

Beyond experimental design, BIBDs are also used in applications of analysis of variance (ANOVA), but more importantly they are at the base of error correcting codes.

3.1 Introduction to Error Correcting Codes

Error correcting codes (ECCs) are codes that make use of several error correcting techniques (with the most common being forward error correction). The main

goal is that given noisy or unreliable communication channels, messages can still be received and read without the entire encoded message being transmitted. This shortens communication time in several cases. One such case is if the communication channel can only move in one direction at a time. This means that if there are two people, Person 1 and Person 2, and Person 1 sends a message to Person 2 along the communication channel, then Person 2 must wait to receive the message before they reply. If Person 2 attempts to send a message to Person 1 while Person 1 is also sending a message to Person 2, then the signals will collide in the communication line and neither person will receive the message intended for them. Another use is with deep space probes in NASA. The communication channel may be two-way (can send and receive at the same time), but the probes are so far from Earth that if part of a message is lost in translation and NASA needs to wait for a new response, NASA now needs to wait even longer to receive a message they could have had ages before. Thus, if NASA has a way to determine what was lost in the message, there is no need to wait for the probe to resend the message, and thus waiting time is greatly reduced [2].

3.2 BIBDs and Error Correcting Codes

The role that BIBDs play in ECCs is not apparent when looking at the code directly, even though they are at the base of many common forward error correction codes. One way that forward ECCs can be written is by using block codes, where block codes encode data into blocks of discrete sizes [4]. This is an immediate connection to BIBDs. Block codes take data of a certain size (containing a certain number of vertices), and stores them into blocks of known size. Thus, if we were attempting to make a BIBD we would already have k, b, and v defined. From here we would only need λ and we could construct a BIBD

using Theorems 2.2 and 2.3. There are many different types of block codes out there (with each essentially describing its own parameters); many of which are taken for granted on a daily basis by the majority of the world. One such class of block codes are the Reed-Solomon (RS) codes. RS codes permeate our current modern world. These codes were first developed in 1960 by Irving S. Reed and Gustave Solomon, and have found use in everything from deep space communication to consumer electronics. This includes, but is definitely not limited to, CDs, DVDs, Blu-ray discs, WiMAX data transmission for cellular devices, ATSC television broadcasting, and RAID6 data backup systems [3]. All of these objects and concepts would cease to exist if not for block designs and BIBDs. In addition, this is just one of many examples of a class of codes that utilized block designs. Without any of these classes of codes that take advantage of block designs, modern electronic communication would be handled extremely differently at an electronic level. Therefore, as we sit in front of our computers, or read emails on our phones, we are using BIBDs and block designs without even realizing it. This speaks volumes to use and versatility provided by BIBDs, even if their ubiquity is often overlooked.

References

[1] M. Bona. *A Walk Through Combinatorics: An Introduction to Enumeration and Graph Theory*. World Scientific Publishing Company Incorporated, 2011.

[2] Diana Johnson Charles Wang, Dean Sklar. Forward error-correction coding. *Crosslink*, page 4, Winter 2002.

[3] Barry A. Cipra. The ubiquitous reed–solomon codes. *SIAM*, 26(1), January 1993.

[4] J. H. van Lint. *Introduction to coding theory*. Springer-Verlag, Berlin, second edition, 1992.