Physical Redundancy for Defect Tolerance: Example Designs and Fundamental Limits

by

Jennifer Tang

B.S.E, Princeton University (2013)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2015

Author ..... Signature redacted
Department of Electrical Engineering and Computer Science
August 28, 2015

Certified by ..... Signature redacted
Yury Polyanskiy
Associate Professor
Thesis Supervisor

Accepted by ..... Signature redacted
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Theses

Physical Redundancy for Defect Tolerance: Example Designs and Fundamental Limits

by
Jennifer Tang

Submitted to the Department of Electrical Engineering and Computer Science
on August 28, 2015, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

This project analyzes designs for physical redundancy which are modeled abstractly as a bipartite graph. The goal is to determine the characteristics of graph structures which optimize the trade-off between the number of edges and the number of redundant components or nodes needed while correcting a deterministic number of worst-case errors. This thesis looks at finite-sized designs, asymptotically large designs with finite error correcting values, and designs with asymptotically large error correcting values. Results include some small optimal graph structures and fundamental limits on what the optimal design structure can achieve for the cases where a small number of errors are corrected and for where the number of errors to be correctly grows asymptotically.

Thesis Supervisor: Yury Polyanskiy
Title: Associate Professor

Acknowledgments

I would like to acknowledge my thesis supervisor Yury Polyanskiy for his support and guidance throughout this research project.

I would like to thank Da Wang for starting this project and giving me helpful advice.

I would also like to thank my family and friends for their encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Information theory has developed a very mature model for adding redundancy to data. In particular, there is an understanding of how to design systems so that if data sent is corrupted by certain noise sources, the original message can still be received with little or no error. This idea along with others from Shannon set a standard for developing algorithms for building redundancy into data. Since the formulation of information theory, there has been significant progress in the both the practical and theoretical understanding of how to design robust information for transmission.

However, these developments are confined to information. There is no such in-depth model when it comes to adding redundancy for physical objects. What are the fundamental rates that can be achieved given constraints on the number of errors to be corrected? What are the structural designs we can use for replacing broken components of a larger system, which are analogous to Hamming codes for transmitting data? These are the types of questions which prompted this project. Our overarching goal is to be able to develop a theory of redundancy for replacing tangible objects.

In this thesis, we will look at finding optimal structures for physical redundancy, taking a first step towards a general theory. The details of this model are given in Chapter 2. We will first give some motivation for this model.

1.1 Motivation

1.1.1 Reconfigurable Circuits

We refer to reconfigurable circuits as a device where multiple stages of configuration are used to build the circuit. First, the circuit is created where only spaces for circuit parts and interconnections between spaces for circuit parts is fabricated. Once an application is decided, the circuit undergoes the first stage of configuration where the specific components needed are chosen for each space. Such a scenario is similar to what is needed to program an FPGA (field-programmable gate array). The look-up tables and wiring is what the FPGA comes with. The first stage configuration selects which values go into which look-up tables.

After this configuration stage, it is possible that some of the configured components have defects, which can occur from contamination, process variation, and

material defects. A single mistake in one component can completely undermine the function of the entire circuit. One way to solve this is to add redundancy to the design. Suppose that extra spaces are allocated as redundant spaces. While configuring the components for the original circuit, some choice of redundant components is also chosen for the redundant spaces. Then after the first stage of configuration, we can test the circuit to determine where the defective components are. Afterwards, we can run a second stage of the configuration, where the redundant components replace any components with defects.

In such a design, it is not feasible for every redundant component to be able to replace every functional component. Wiring needs to exist between spaces for this to occur. What then, is the optimal way to design the redundant spaces and its wiring?

This question was the original motivation for our exploration of physical redundancy. However, we do not wish to confine the model developed in this project only to reconfigurable circuits. The next motivation illustrates how other problems can also benefit from the analysis of redundancy systems.

1.1.2 Server Farms

Consider a network of server farm designed to be sold to a client. The servers available are very basic. Once the client purchases a network of k servers, each server is configured to perform one of q functions. However, it is possible that some of the servers are down at certain moments of operation. Thus, the k servers also come with an additional m redundant servers which can be used to pick up the work of a server which is not operating fully. Each of these redundant servers must also be configured by the client to one of the q functions before operation. Infrastructure needed to connect the m redundant servers to the original comes at a cost. As the designer of the complete network of $k+m$ servers, what is the best way to make the most efficient connections between the redundant servers and original servers?

1.1.3 Graph Coloring

Imagine that you have a bipartite graph G. G has k left-side nodes and m right-side nodes. Node can be colored one color from the set $\mathcal{X}$. What is the minimum number of edges necessary for G, so that for any coloring of the k left-side nodes, there exists a coloring for the m right-side nodes so that each left-side node has t neighbors with the same color as itself?

While this problem in graph coloring is not about redundancy, it has everything to do with the question we are trying to solve. We list this graph coloring problem as a motivation because we just also interested in solving this problem for the sake of combinatorics. This motivation can be seen as a pursuit of inner beauty.

1.2 Overview of Main Contributions

In this thesis, we developed a simple model for physical redundancy. The model is a bipartite graph which we call a design. Each design consists of

- Functional nodes
- Redundant nodes
- Inconnections between the functional nodes and redundant nodes

The main goal is to determine what are the trade-offs in wiring complexity (average degree) and number of redundant nodes for designs that correct t number of worst-case errors simultaneously for any functional node realization.

Given this model, we have determined the following results in this thesis:

- Optimal designs on small number of nodes
- Fundamental limits and optimal designs on small alphabet sizes and small error correcting values t
- Fundamental limits on all designs and optimal designs when t is asymptotically large.

The optimal small designs can be used practically for physical redundancy applications if these applications meet the parameters of the design.

The fundamental limits for small error correcting values tells us importantly that the optimal designs are those which are a linear combination of two basic designs. These basic designs are when:

- All redundant nodes are each dedicated to correcting only one functional node
- All redundant nodes can correct errors for any functional node

Combinations of these two designs are optimal for large designs correcting small errors.

The fundamental limits for asymptotic t defines all possible trade-offs any design can have. It expresses the bound which no design can surpass. The results also tell as that the asymptotically best designs are ones which are symmetric graphs.

1.3 Related Work

1.3.1 The Theoretical: A Design for Fault Tolerance

The the best of out knowledge, no has has studied abstract physical redundancy models in the way we have developed in the thesis. However, there has been previous work on adding redundancy to for many circuit applications.

Von Neumann is known to have made the first contribution to the area of designing redundancy for circuit applications [1]. He explored the problem of how to correct faulty signals coming from noisy gates. Since von Neumann was connecting his work to the functions of the human nervous system, he used terminology from biology, and describes what we know as gates to be organs. The main idea behind von Neumann's construction is that the error probability of a faulty organ can reduced when the single organ is replaced with a network of multiple organs and single wire lines are replaced with a bundle of wires. The two parts to his construction are called the executive organ and the restoring organ. The executive organ consists of several copies of the desired organ. The output signals of these copies undergo a permutation before entering the restoring organ, which are a set of organs that take a majority vote.

The unfortunate aspect of von Neumann's design is that in order to be effective in correcting errors, a very large number of lines and organs are needed, which makes the design not practical. Nevertheless, von Neumann's work is very influential and commonly referenced. Von Neumann's ideas inspired even Shannon, who co-authored a paper with Moore on designing a redundancy system for reliable relays [3].

One major follow-up on von Neumann's paper is by Pippenger, who proved results on how many noisy redundant gates are needed asymptotically to achieve certain functions [2]. Pippenger's main construction follows von Neumann's ideas. He uses a concept called a compressor graph to permute wires more precisely, and he has a hierarchy structure in many of his constructions. However, overall, Pippenger's construction, while only needing a linear number of gates asymptotically, still requires too many components to be practical. For instance, one construction of Pippenger's requires 8^{17} elements. Yet, von Neumann's and Pippenger's designs are important to many theoretical work on redundancy.

1.3.2 The Practical: A Design for Defect Tolerance

The work started by von Neumann focuses on fault tolerance, fixing mistakes which occur in one instance of some calculation. These are sometimes called transient errors. Another line of work focuses on defect tolerance, which involves physical errors permanently present in the device. Von Neumann's design can also used to fix defects, but this was not the primary goal of the original construction. A well-known project which specifically aims at correcting defects Teramac. Teramac is a computer build by HP labs [4]. The goal of Teramac is to build a computer using FGPAs with defects in them by reconfiguring around the defects. Researchers were able to do this by implementing a hierarchical design. Each level of devices can communicate with adjacent levels using a fat tree structures. The benefit of this kind of structure is that any two nodes have many possible communication connections between them.

The Teramac project concluded importantly that it is possible to build a powerful computer from defective parts. However, this comes at the expense of needing to test the machine for defects and then using an algorithm to reconfigure it. Teramac needed a large number of wires in order to be versatile enough so that finding a configuration is possible.

Another fault tolerant technique worth mentioning is the Network-on-Chip design

used in VLSI circuits. This design lays out small IP cores in a grid structure. Each IPs does its own computation, and can communicate data to adjacent IPs. A type of algorithm called randomized gossip protocols have been researched on this type of network [5]. These are interesting in that the randomization reduces the communication bandwidth, yet still spreads data exponentially quickly. Experimental simulations have been conducted on this idea to show that this idea has good average case behavior.

Some theoretical work has also been done on the Network-on-Chip design by Leighton who analyzed procedure to wiring around faults which result in the shortest wire length [6].

1.3.3 Experimental Comparisons

Researchers study the designs and techniques created by von Neumann and Teramac in order to find solutions for fixing defects in nanotechnology. There has been work done on experimental simulations which compare these different structures for defect tolerance in terms of amount of redundancy.

One such work models how much redundancy is needed for a given amount of reliability in von Neumann's model [8]. In this project, probability model checking is used and results were shown that adding more restorative stages is effective for small probability of failures, but not effective when the gate failure probability is large.

Another interesting work compares the reliability and redundancy of von Neumann designs, with the reconfiguration design, which is based on the design of Teramac, and modular redundancy [7]. (Modular redundancy is a design where a single unit is replaced by three copies of the unit, and a majority gate. The output of the majority is the output of the system.) This work theoretically models the behavior of each design to compare their performances. The result is that reconfiguration is able to handle larger defect rates, though the amount of redundancy will become extremely large. In general though, reconfiguration performs better than modular redundancy and von Neumann's technique, even if there are still drawbacks to the design.

This experimental result that reconfiguration performs better on the reliability verses redundancy trade-off for small amount of defects influences our line of research. The major drawback to the reconfiguration design is that it uses a large amount of wires. Our work deals with the questions of how well reconfiguration does on the wire complexity and redundancy trade-off, which is a necessary step in the direction of evaluating reconfiguration's overall usefulness as a redundancy technique.

While Teramac uses a hierarchical structure, we want to simplify matters in our model, and only consider one level of redundancy.

1.4 Organization of this Thesis

Chapter 2 starts by explaining the model we developed for analyzing physical redundancy. Main results are discussed in Chapter 3. In Chapter 4 we go over results for finite number of nodes. This is similar to the problem of finding a structure for

physical redundancy equivalent to what the Hamming Codes are for information. We develop some tools for our analysis in Chapter 5. In Chapter 6 we describe results for asymptotically many components with finite error correcting value. In Chapter 7 we describes results for all asymptotically large quantities, which characterizes a fundamental limit of the wiring complexity and redundancy trade-off. We end with a conclusion in Chapter 8.

Chapter 2

Model Setup

This chapter defines the model and the notation used. We will also provide explanation for the choices used in our model.

2.1 Model Specification

A **design** G is a bipartite graph $G = (u^k, v^m, \mathcal{E})$, where $u^k = \{u_1, u_2, ...u_k\}$, $v^m = \{v_1, v_2, ...v_m\}$ and $\mathcal{E}$ is the set of edges between nodes in u^k and v^k.

$u_1, u_2, ..., u_k$ are called the **functional nodes** and $v_1, v_2, ..., v_m$ are the redundant **redundant nodes.** Let $E = |\mathcal{E}|$

Functional nodes and redundant nodes can each be labeled a value in the set of elements $\mathcal{X}$. For simplicity, we usually let $\mathcal{X} = \{0, 1, ..., L-1\}$. We assume that $L \geq 2$ and refer to L or $|\mathcal{X}|$ as the alphabet size.

A labeling of functional nodes are a k-tuple of values $s^k = s_1, s_2, ..., s_k$ where each $s_i \in \mathcal{X}$ and functional node u_i is labeled s_i. Similarly, a labeling of redundant nodes are a m-tuple of values $r^m = r_1, r_2, ..., r_m$.

Definition 1. *A bipartite graph is called a (k, m, t, E)-design if*

- *the number of functional nodes is k*
- *the number of redundant nodes is m*
- *the number of edges is E*
- *for every labeling $s^k \in \mathcal{X}^k$ given to the functional nodes, there exists a labeling $r^m \in \mathcal{X}^m$ of the redundant nodes, so that for every functional node u, if u is labeled $x \in \mathcal{X}$, then u has at least t neighbors also labeled x.*

We are primarily interested in finding values of m and E for which there exists a (k, m, t, E)-design normalized by k. This is captured by

- **Redundancy** of a (k, m, t, E)-design is $\rho = \frac{m}{k}$
- **Wiring complexity** of a (k, m, t, E)-design is $\varepsilon = \frac{E}{k}$

Definition 2. *For a fixed $\mathcal{X}$ and $t \geq 1$ we define the region $\mathcal{R}_t$ as the closure of the set of all achievable pairs of (ε, ρ):*

$$\mathcal{R}_t \triangleq \text{closure}\left\{(\frac{E}{k}, \frac{m}{k}) : \exists(k, m, t, E) - \textit{design}\right\}$$

Definition 3.

$$\mathcal{R}_\infty \triangleq \text{closure}\{(\frac{m}{kt}, \frac{E}{kt}) : \exists(k, m, t, E) - \textit{design}\}$$

Decisions for creating this model are explained in the following sections.

2.2 Physical Interpretation of Model

The functional nodes represent an objects which need to either have no errors or to be replaced by an object which has no errors. These nodes can represent any object, let it be circuit components, servers, graph vertices, or crayons. A redundant nodes an object which can be used to replace functional nodes in an event the functional has an error.

Both functional nodes and redundant nodes can have errors. A redundant node with an error is unable to replace a functional node. Errors in the model can represent defective, broken, or unavailable objects.

It is unknown what kind of object the functional node is (or will be) until the functional node is labeled. Both functional nodes and redundant nodes are given a label $x \in \mathcal{X}$.

Elements in $\mathcal{X}$ can represent any set of things. Examples of $\mathcal{X}$ could be

- gates such as NAND gates or NOR gates
- different instances of look-up tables
- Server functions
- Colors

The significance of $\mathcal{X}$ is that functional nodes with label x can only be replaced by redundant nodes of the same label x. After all, it does not make sense to replace a broken NAND gate with a NOR gate. As stated in Section 2.1, for simplicity we usual let $\mathcal{X}$ to a set of numbers.

If any redundant nodes are able to replace any functional nodes, this situation woud not be interesting to analyze. Also, this does not reflect real world constraints. In a circuit, only so many wire connections exist, so not every redundant component can replace every functional component. There may also be constraints due to proximity.

When a specific redundant node is able to replace a specific functional node, we recognize that by connecting the two nodes with an edge e. If a functional node and a

redundant node are connected by an edge e, then if the functional node has an error, the redundant node can replace it if the redundant node does not have an error and is not already used to replace another functional node. In the circuit analog, the edge e between the functional node and redundant node behaves as a wire.

There are certainly other models we could have chosen for a redundancy system, but this one is selected due to its simplicity.

2.3 Metrics

To compare the performance of one design against another, we choose metrics which favors using fewer resources. There are two types of resources we use in our model, number of redundant nodes and number of edges. This justifies the redundancy and wiring complexity metrics.

- **Redundancy**. Denoted by ρ, this is the ratio of number of redundant nodes to number of functional nodes in a given design. $\rho = \frac{m}{k}$.
- **Wiring complexity**. Denoted by ε, this is the ratio of number of edges to number of functional nodes in a designs. This value also represents the average degree of functional nodes. $\varepsilon = \frac{E}{k}$

These two metrics will be the primarily focus of this research. Other metrics which may be important for applications in redundancy include like physical layout or maximum degree of nodes, but we will not consider these for this project. Another metric which may be of interest is regularity, where the functional nodes (or perhaps the redundant nodes) all have similar degree.

There is a natural trade-off between minimizing amount of redundancy and minimizing number of edges. A design that uses more redundant nodes can get away with fewer edges, whereas a design that uses fewer redundant nodes needs to have more edges in order to be t error correcting. This is illustrated in Chapter 4.

For each t, we can plot ε and ρ for all possible (k, m, t, E)-designs on the redundancy and wiring complexity plot (we will call these the ε, ρ plot) to see what point of the trade-off it achieves. The achievable region is defined as the set of all ε and ρ pairs a t error correcting design can have. This is what is defined to be R_t.

2.4 Error Correcting Method

When correcting errors, we need to specify in what sense the errors are corrected. Two possibilities are:

- **Probabilistic**. Each node fails with some probability p. The goal of the design is to be able to correct all the errors with some probability.
- **Deterministic**. The case corrects for worst-case errors. The goal of the design is to be able to correct all possible sets of t errors

While both cases are interesting, for the purposes of this thesis, we will focus only on deterministic error correction. When we say a design with a labeling s^k and r^m is t error correcting, we mean that for any t nodes with errors (recall these can be functional nodes or redundant nodes), there is a way for every functional node u which has an error to be replaced by a redundant node. The criteria for a redundant node to replace a functional node is that it must:

- share can edge with functional node u
- have the same label as functional node u
- not already be replacing another functional node with an error

There is a straight forward criteria to check if a design is t error correcting.

Proposition 1. *A design G is t error correcting if and only if every functional node labeled the value $x \in \mathcal{X}$, has t or more neighboring redundant node also with value x.*

Definition 4. *Let the **effective neighbors** of u be the set of neighbors of node u which has the same value as u.*

Effective neighbors can only be determined once both functional nodes and redundant nodes are labeled.

To be clear on our terminology, when we refer to a design G as t error correcting it could mean one of the following:

- Design G is t error correcting means there exists r^m for each s^k, where each functional node has t neighbors with the same value as itself
- Design G with labeling s^k is t error correcting means there exists r^m for this specific s^k, where each functional node has t neighbors with the same value as itself
- Design G with labeling s^k and r^m is t error correcting means for this choice of s^k and r^m, each functional node has t neighbors with the same value as itself

The fact that there is so many ways to define error correcting clearly calls for a need to further define the problem. The next section considers these scenarios.

2.5 Design Scenarios

The design decision in Definition 1 to find structures which are t-correcting for all $s^k \in \mathcal{X}^k$ was chosen because it is the most interesting of the possible scenarios.

Here are 3 scenarios we can design for:

- **Scenario A: Given s^k, design G and r^m.** The interconnects and labeling of r^m are designed for a specific s^k that is known.

- **Scenario B: Design G, then given s^k , design r^m.** Design G is structured in a way such that for all $s^k \in \mathcal{X}^k$, there is some way after s^k is specified to label r^m.
- **Scenario C: Design G and r^m when s^k is unknown.** This design fixes G and redundant labeling r^m. There is no knowledge of s^k.

Designs which correct errors in scenario C, also work for scenario B, and those that work for scenario B work for scenario A. It turns out the the optimal designs in terms of the redundancy and wire complexity metrics for scenario A and scenario C are not very interesting.

Design A Performance

For scenario A, the optimal design is one where each functional node has t edges that connect to t set of redundant nodes with the same value.

To see this, first note that each functional node must have at least t edges. At minimum, there has to be at least t redundant nodes of each element. If each functional node with value x is connected to all t redundant nodes of value x, this design is t error correcting and meets the minimum number of edges and redundant nodes requirement. Only a finite number of redundant nodes are needed, so as $k \to \infty$, the amount of redundancy $\rho = \frac{m}{k}$ goes to 0. Thus, this is the optimal design.

Design C Performance

For scenario C, the optimal design is one where each functional node has $|\mathcal{X}|t$ edges, each connected to a t redundant nodes for each $|\mathcal{X}|$. The proof for this is similar to be proof for scenario A. Each functional can take on one of $|\mathcal{X}|$ values. Since all the redundant nodes are already fixed, all nodes must to connected to t copies of each value in $\mathcal{X}$. The minimum number of redundant nodes occurs when all functional nodes are connected to the same set of redundant nodes.

Performance Comparison

Define a $(k, m, t, E) -$ A-design similarly to a $(k, m, t, E) -$ design in Definition 1 except with the condition that $\exists s^k$ where the design for some r^m is so that each functional node has t neighbors with the same value as itself.

Define a $(k, m, t, E) -$ C-design similarly to a $(k, m, t, E) -$ design in Definition 1 except with the condition that the design is so that $\exists r^m$ where $\forall s^k$ each functional node has t neighbors with the same value as itself

Let

$$\mathcal{R}_t^A \triangleq \text{closure}\left\{(\frac{m}{k}, \frac{E}{k}) : \exists (k, m, t, E) - \text{A-design}\right\}$$

$$\mathcal{R}_t^C \triangleq \text{closure}\left\{(\frac{m}{k}, \frac{E}{k}) : \exists (k, m, t, E) - \text{C-design}\right\}$$

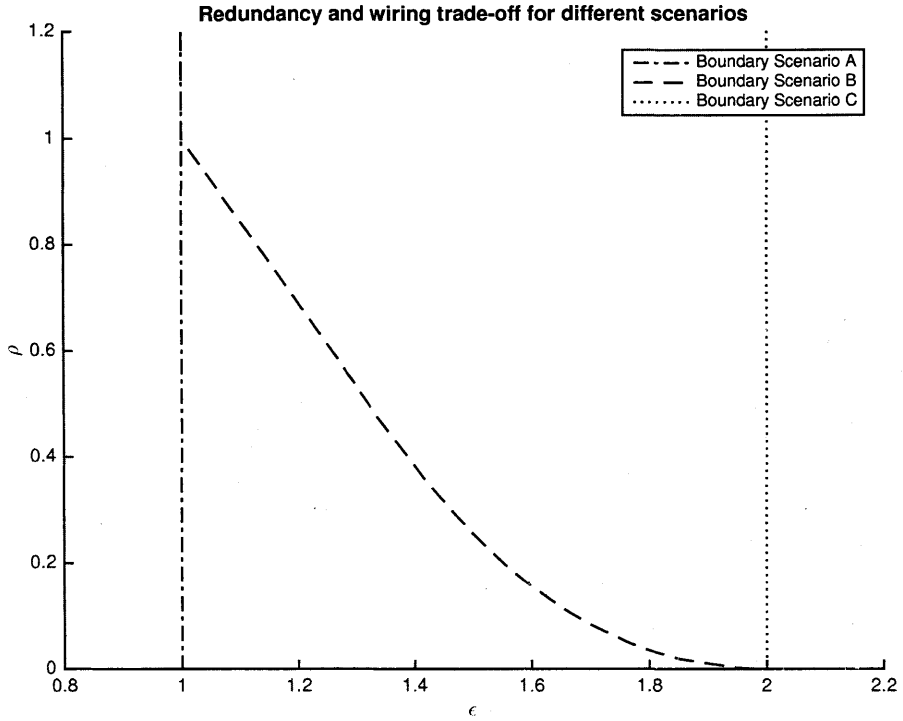


Figure 2-1: Plot comparing achievable region for Scenarios A, B and C. Lines plotted are the boundary of the different regions.

We can plot the achievable region $\mathcal{R}_t^A$ for scenario A and $\mathcal{R}_t^C$ for scenario C on the ε, ρ plot. These both are bounded by vertical lines, since ε is limited by the number of neighbors each node has to have, while $\rho \to 0$ as $k \to \infty$.

Finding the solution for scenario B, on the other hand, is not trivial. It is not equivalent to the performance of Scenario A or C, but it lies somewhere in-between. A plot of this is shown in Figure 2-1.

Focusing on scenario B also lets the problem become one about structures. The answer the questions which bipartite graphs are good for redundancy, regardless of the types of objects in the graph. A good design will need to simultaneously t error correcting for all choices of labeling.

Focusing on scenario B for the remainder of the thesis justifies Definition 1.

2.6 Alternate Definitions: Hypergraph

An alternative way of defining our problem is in terms of graph coloring. Suppose we have a hypergraph. We want to design the hyperedges of the hypergraph such that for any labeling of colors for vertices, where each color is in $\mathcal{X}$, there is always a way to color the hyperedges so that each vertex has t edges with the same color as itself.

Define

$d_t(k, m) = \min(\text{average edge-size: all } (t, t) \text{ colorable hypergraphs on } m \text{ vertices and } k \text{ hyperedges})$

In this definition of the problem, the vertices correspond to functional nodes and the hyperedges correspond to redundant nodes.

Lemma 1. *The boundary of* $\mathcal{R}_t$ *is given by* $\lim_{k\to\infty} d_t(k, \lceil \rho k \rceil)$.

Chapter 3

Summary of Main Results

The first main results of this paper finds $\mathcal{R}_t$ for small values of t.

Theorem 1. *When $|\mathcal{X}| = 2$, for $t = 1$ and $t = 2$, and $|\mathcal{X}| = 3$, for $t = 1$,*

$$\mathcal{R}_t = \{(\varepsilon, \rho) : \varepsilon \geq t \text{ and } \varepsilon \geq |\mathcal{X}|t - (|\mathcal{X}| - 1)\rho\}$$

This is discussed in Section 6.

The second main theorem is the definition of $\mathcal{R}_\infty$, the achievable region of designs when $t \to \infty$. This region is defined in terms of an optimization function for each parameter $\mathbb{E}[S]$.

Theorem 2 (Asymptotic t Result). *When $|\mathcal{X}| = 2$, the region $\mathcal{R}_\infty$ as defined in Definition 3 is the closure of points $(\tilde{\varepsilon}, \tilde{\rho})$ parameterized by P_S on $\mathbb{Z}_+$ with finite support and*

$$\tilde{\varepsilon} = \frac{\mathbb{E}[S]}{F(P_S)}, \tilde{\rho} = \frac{1}{F(P_S)},$$

$$F(P_S) \triangleq \min_{\lambda \in [0,1]} \max_{0 \leq f_{l_0,l_1} \leq 1} \min \left\{ \mathbb{E}\left[\frac{L_0}{\lambda} f(L_0, L_1)\right], \mathbb{E}\left[\frac{L_1}{1-\lambda}(1 - f(L_0, L_1))\right] \right\} \quad (3.1)$$

where the expectations are taken over $S \sim P_S$ and given S the distribution of $L_0 \sim Bino(S, \lambda)$ and $L_1 = S - L_0$.

This is discussed in Section 7. The achievable regions for both theorems are plotted in Figure 3-1.

The result for $\mathcal{R}_1$ and $\mathcal{R}_2$ demonstrates that for correcting small errors, the best solution in the limit of a large number of functional nodes is a linear combination of two basic designs, one where each redundant node is connected to only one functional node and one where each redundant node is connected to all functional nodes. [1] Theorem 1 tells us that for any k no design can do better than a linear trade-off

[1] Though this design is not optimal for finite k. Slight improvements can be made to this basic design.

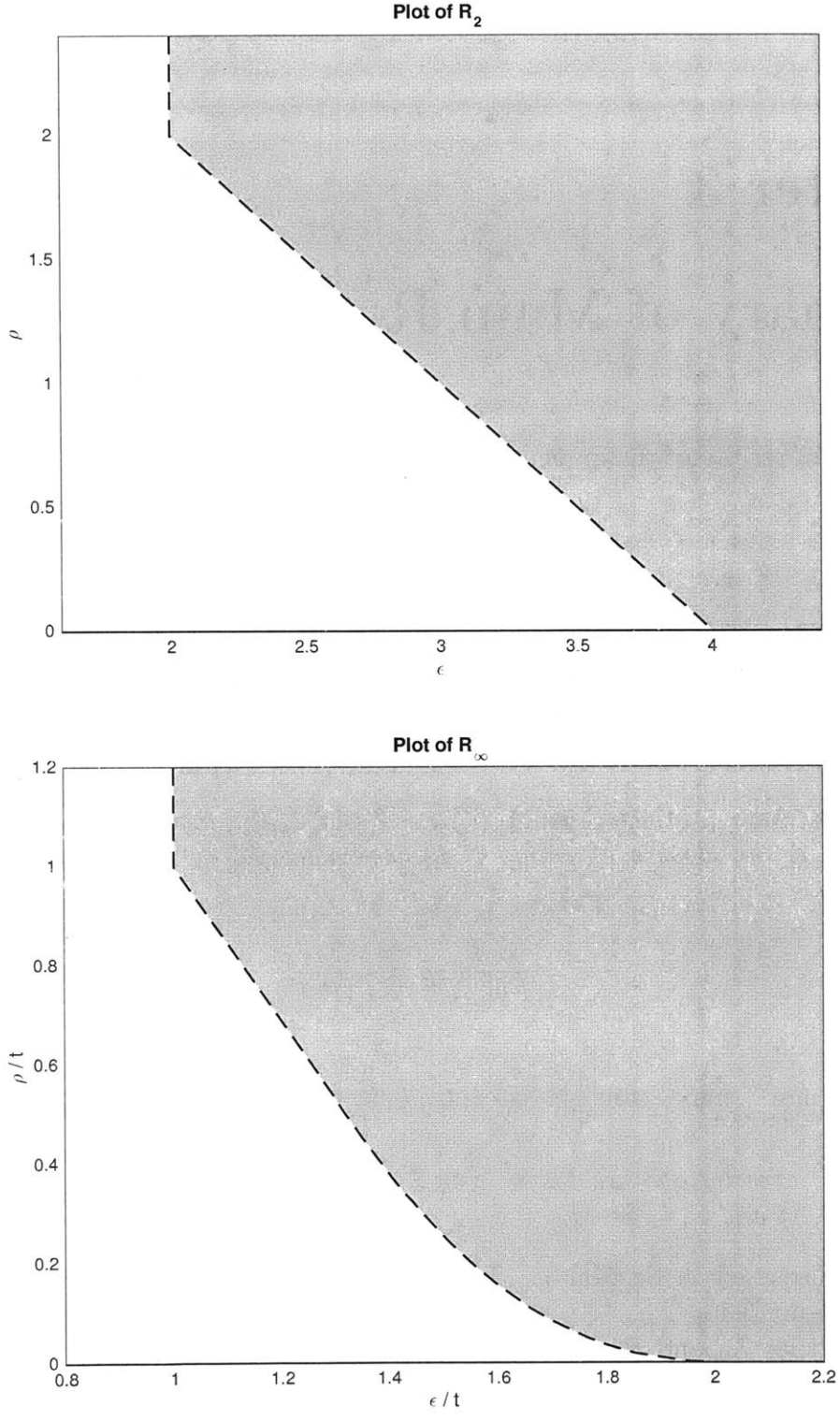


Figure 3-1: Plots for $\mathcal{R}_2$ and $\mathcal{R}_\infty$. The shaded area represent achievable region. The plot for $\mathcal{R}_\infty$ is normalized by the number of errors corrected. The boundary of the region is calculated with an approximation.

between wiring complexity and redundancy for small error correcting values of 1 and 2.

It is unknown what exactly the region for $t > 2$ will look like, but the asymptotic result Theorem 2 captures what the regions will approach as $t \to \infty$. All regions $\frac{1}{t}\mathcal{R}_t$ will lie between $\mathcal{R}_1$ and $\mathcal{R}_\infty$. Thus, the boundary for $\mathcal{R}_\infty$ also acts as a lower bound for all other region $\mathcal{R}_t$. Regardless of the design structure, the tradeoff between redundancy and wiring complexity cannot surpass the fundamental limit given by $\mathcal{R}_\infty$. Theorem 2 tells us that the design which asymptotically achieve the best trade-off are symmetric graphs.

Chapter 4

Examples: Finite k

In this section, we will discuss examples of designs and their performance on the ε, ρ plot. These examples show what kind of trade-offs exist for designs with small k. Some of the examples are pivotal for proving our major theorems.

4.1 Basic Designs

4.1.1 Repetition Block

One basic design is where each redundant node is dedicated to only one functional node. These are called repetition blocks. They are a family of (k, tk, t, tk)-designs. We will refer to these redundant nodes which are only connected to one functional node as **private nodes**. See Figure 4-1.

Definition 5 (Repetition block). *$K(1,t)$ is the repetition block on where one functional node is connected to t private redundant nodes.*

The repetition block of k functional nodes is written as $kK(1,t)$. This designs shows that the point (t,t) is achievable in $\mathcal{R}_t$. It is straightforward to see that if $\varepsilon = \frac{E}{k} = t$, then $\rho = mk = t$ is the smallest that is possible. Since no design can have $\varepsilon < t$, the repetition is then the optimal design for minimizing wiring complexity. Yet, it uses the worst amount of redundancy possible.

Note that the structure of this design does not depend on the alphabet size.

4.1.2 Complete Design

Another basic design is the complete design. The complete design has every functional node connected to every redundant node. See Figure 4-2.

The complete graph is a family of $(k, t|\mathcal{X}|, t, kt|\mathcal{X}|)$-designs for different values of t and k. This design works by labeling for each value x in $\mathcal{X}$, t number of redundant nodes to the value x. This way, every functional node will be connected to t redundant nodes of each value, so each functional node must have t effective neighbors.

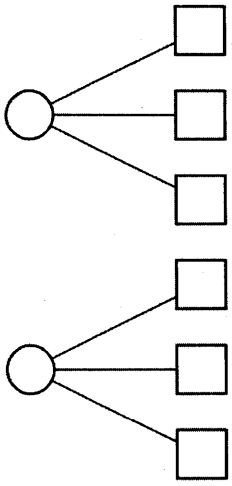


Figure 4-1: The block design. This is an example of a $(2, 6, 3, 6)$-design.

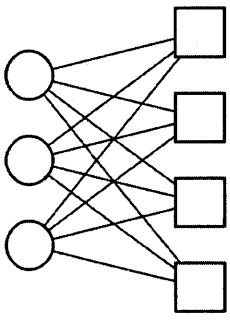


Figure 4-2: The complete design. This is an example of a $(3, 4, 2, 12)$-design.

Definition 6 (Complete design). *$K(k, m)$ is the complete design on k functional nodes and m redundant nodes.*

For this design, $\varepsilon = t|\mathcal{X}|$ and $\rho = \frac{t|\mathcal{X}|}{k}$. An important feature of the complete graph is that as $k \to \infty$, the number of redundant nodes do not have to change, which causes $\rho \to 0$.

For finite k, however, the complete graph is not the optimal design given the amount of redundancy. It is possible to remove some of the edges and still maintain a t error correcting design, as we will show in the next example.

4.2 Optimal Small Designs

In this section we will look at some examples which fixing the parameters k, m, t are optimal in E.

4.2.1 Hamming Block

The Hamming block is a $(3, 4, 2, 9)$-design. Each pair of functional nodes are connected by a redundant node and there is also a redundant node connected to all three functional nodes. The Hamming block is specifically designed to be a 2 error correcting design when $|\mathcal{X}| = 2$. See Figure 4-4(a).

The Hamming block has $\varepsilon = 3$ and $\rho = \frac{4}{3}$, outperforming the complete complete graph with the same number of functional nodes.

In fact, it turns out that for $k = 3$, $m = 4$, $t = 2$ and $|\mathcal{X}| = 2$, the Hamming block is the unique optimal design in terms of minimizing the number of edges. It is particularly nice in that does not use too many edges or have too many redundant nodes and has the nice feature of being symmetric.

4.2.2 Optimal Designs for Various Alphabet Sizes

Some small optimal designs are not interesting. For instance, when $k \leq |\mathcal{X}|$, the best design simply the repetition block $kK(1, t)$. The first interesting cases occur when $k = |\mathcal{X}| + 1$. In this case, the smallest number of redundant nodes is $m = t|\mathcal{X}|$. The Hamming block was the first example of this discovered. Other designs were later found through a combination of experimentation and analysis.

The optimal designs with for $t = 1$ are:

- $|\mathcal{X}| = 2$, k = 3, m = 2 with $E = 5$. See Figure 4-3(a)
- $|\mathcal{X}| = 3$, k = 4, m = 3 with $E = 8$. See Figure 4-3(b)
- $|\mathcal{X}| = 4$, k = 5, m = 4 with $E = 12$. See Figure 4-3(c)

The optimal designs with for $t = 2$ are:

- $|\mathcal{X}| = 2$, k = 3, m = 4 with $E = 9$. (Hamming block) See Figure 4-4(a)

- $|\mathcal{X}| = 3$, k = 4, m = 6 with $E = 15$. See Figure 4-4(b)
- $|\mathcal{X}| = 4$, k = 5, m = 8 with $E = 21$. See Figure 4-4(c)

It should be noted that some of these designs are uniquely optimal. For instance, the Hamming block and the design for $|\mathcal{X}| = 2$ and $t = 1$ pictured in Figure 4-3(a) are uniquely optimal. However, it is known that for both of the $|\mathcal{X}| = 3$ designs, there are two different designs which are both optimal. We can find at least 4 optimal $|\mathcal{X}| = 4$ and $t = 1$ designs.

For $|\mathcal{X}| = 2$, there exists a family of $(k, 2(k-1), k-1, 2(k-1)k-k)$-designs. These are optimal for in the number of edges when the other parameters are fixed. The designs that accomplish this are a merging of subset designs $S(k, k-1) \vee \left(\vee_{j=1}^{k-2} S(k,k)\right)$ (See Section 4.3 for these definitions.) This design can also be understood as a complete design with a k edges removed. The Hamming block belongs to this family.

4.3 Subset Designs

We will present the development of design we refer to as subset designs[1] since they will be important results in Chapter 7.

A subset design is a design where all subsets of functional nodes of a given size are connected together by a redundant node.

To be more precise, subset design $S(k, s)$ is a bipartite graph with k functional nodes and $\binom{k}{s}$ redundant nodes. Let $s \leq k$ be a number we call the subset size. Each of the $\binom{k}{s}$ s-subsets of $\{1, ..., k\}$ will each correspond with a specific redundant node. All the functional nodes in that s-subset will have an edge with the corresponding redundant node. Note that the degree of each functional node is $\binom{k-1}{s-1}$.

Design $S(k, s)$ is what we call a single-size subset design, since we only have subset size of one number s.

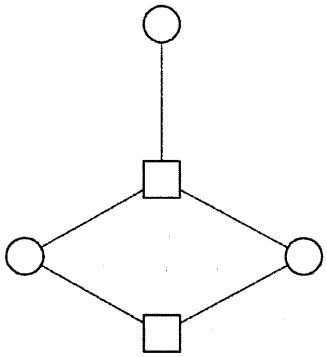
Definition 7 (Single-size subset design). *$S(k, s)$ is the single-size subset design on k functional nodes where each s-subset functional nodes are connected by a redundant node.*

In general we allow subset designs to use multiple and possibly different subset sizes. For two values s_1 and s_2, where $s_1, s_2 \leq k$, the bipartite graph $S(k, s_1) \vee S(k, s_2)$ contains both redundant nodes for all the s_1-subsets and all the s_2-subsets. The resulting graph has a total of $m = \binom{k}{s_1} + \binom{k}{s_2}$ redundant nodes. We call also define $S(k, s_1) \vee S(k, s_2)$ as identifying the k functional nodes in disjoint copies of $S(k, s_1)$ and $S(k, s_2)$[2]. If $s_1 = s_2$, then each s_1-subset is connected by a redundant node twice.

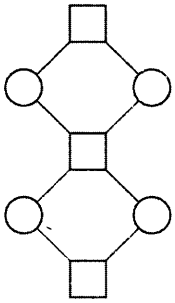
Definition 8 (General subset design). *Given k and positive integers $s_1, s_2, ..., s_r$, the subset design is a bipartite graph $\vee_{j=1}^{r} S(k, s_j) = S(k, s_1) \vee S(k, s_2) \vee ... \vee S(k, s_r)$ with k functional nodes and $m = \sum_{j=1}^{r} \binom{k}{s_j}$ redundant nodes.*

[1] a term we developed for this thesis

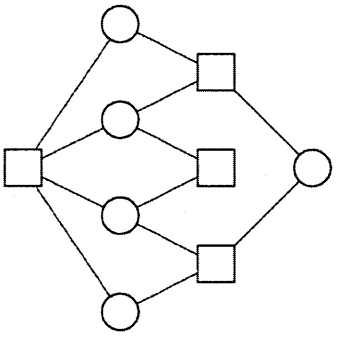
[2] This operation is called merging. This is discussed in Section 5.5.



(a) $|\mathcal{X}| = 2$

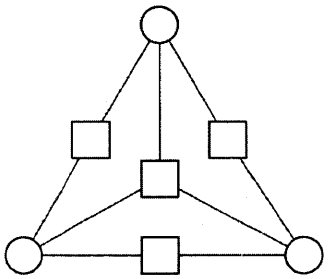


(b) $|\mathcal{X}| = 3$

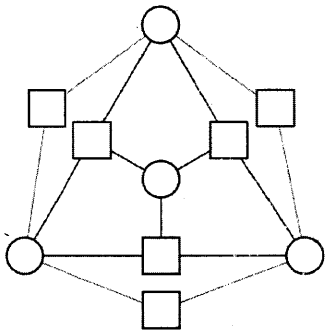


(c) $|\mathcal{X}| = 4$

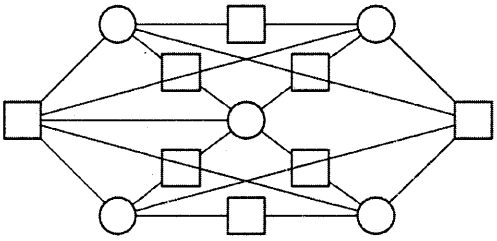
Figure 4-3: Smallest non-trivial 1-error correcting designs.



(a) $|\mathcal{X}| = 2$



(b) $|\mathcal{X}| = 3$



(c) $|\mathcal{X}| = 4$

Figure 4-4: Smallest non-trivial 2-error correcting designs.

Previous examples discussed can also be described as subset designs.

- The Hamming block is a $S(3,2) \vee S(3,3)$ subset designs.
- The repetition block is a $\vee_{j=1}^{t} S(k,1)$ subset designs.
- The complete design is a $\vee_{j=1}^{|\mathcal{X}|t} S(k,k)$ subset designs.

4.3.1 Subset Weight

To specify the subset sizes present in a graph, we will define the weight $P_S(s)$ of each subset size s. To make calculations simpler in later sections, P_S is defined as the proportion of redundant nodes in the entire graph which connect s-subsets. For instance, for the Hamming block $S(3,2) \vee S(3,3)$ there are 3 redundant nodes connecting 2-subsets and 1 redundant node connecting 3-subsets. So the weight of $s_1 = 2$ is $P(2) = .75$ and the weight of $s_2 = 3$ is $P(3) = .25$. If the graph $S(3,2) \vee S(3,3) \vee S(3,3)$, then $P(3) = .4$ and $P(2) = .6$.

4.3.2 Symmetry of Subset Designs

The specialty of subset designs is that they are very symmetric.

Definition 9 (Permuatation Invariance)**.** *A design is called* permutation invariant *if there exists a group of graph automorphisms (preserving functional/redundant node partition) that acts as the full symmetric group S_k on the functional nodes*

Proposition 2. *A design is permutation invariant if and only if it is a subset design.*

Proof. A subset design is clearly preserved under any permutation of its functional or redundant nodes.

If a design is permutation invariant, consider for some $s \geq 1$ and take the subgraph of this design induced by all redundant nodes of degree s and their neighboring functional nodes. (Assume that there is at least one redundant node of degree s).

Let any one s-subset of nodes connected by a redundant node be called the set A. There exists a permutation of this set of A to any arbitrarily s-subset of the k functional nodes. Thus all s-subsets of the k functional nodes must be the neighborhood of some redundant node. The number of times an specific s-subset is connected by a redundant must also be the same for all s-subsets. If each s-subset is the neighborhood of r_s redundant nodes, this corresponds to the subset design $\vee_{j=1}^{r_s} S(k,s)$.

The original design is then expressed as a merging $\vee_{s \in I} \vee_{j=1}^{r_s} S(k,s)$ for a set of subset sizes I. □

4.3.3 Subset Achievability

The number of redundant nodes used in a subset design is unfortunately large. It increases with k to the exponent of the fixed size of the subset s. With so many

redundant nodes, subset designs are able to correct very large t. But for small values of t, subset designs are not a practical way of providing redundancy[3].

Proposition 3 (Subset Achievability). *Let* $m = \binom{k}{s}$. *Then there exists a* (m, k, t, E)*-design with* $E = s\binom{k}{s}$ *and* $t = \Theta(k^{s-1})$.

Proof. We know that $t = O(k^{s-1})$, since each functional node has at most $O(k^{s-1})$ neighbors. To show that $t = \Omega(k^{s-1})$, fix a labeling for the functional nodes.

We will call an element $x \in \mathcal{X}$ rare, if fewer than $\frac{k}{c|\mathcal{X}|}$ functional nodes are labeled x, for some appropriate choice of constant c. We will use the following suboptimal labeling for the redundant nodes:

- a redundant node is labeled x if all its neighbors are labeled x.
- a redundant node is labeled x if there is only one rare element in its neighborhood.
- otherwise, the redundant node can be labeled arbitrarily.

To see that each functional node has $\Omega(k^{s-1})$ effective neighbors, notice that for all functional nodes with non-rare elements, is in at least $\binom{k/c|\mathcal{X}|}{s-1} = \Omega(k^{s-1})$ subsets were all the functional nodes are labeled x. The total number of rare elements does not exceed $\frac{k}{c}$, so each rare element is in at least $\binom{k(1-1/c)}{s-1} = \Omega(k^{s-1})$ subsets where it is the only rare element.

□

Notice that the quantity $\frac{E}{kt}$ is a constant as k grows. We will use this idea to get sharper bounds on t for subset designs in Chapter 7.

[3]unless of course $s = 1$ or $s = k$

Chapter 5

Achievable Regions

The chapter contains the necessary ideas about the achievable regions $\mathcal{R}_t$ and $\mathcal{R}_\infty$ for defining and showing results in Chapter 6 and Chapter 7.

When we say a (k, m, t, E)-design G achieves (ε, ρ) on $\mathcal{R}_t$, we mean that $\varepsilon = \frac{E}{k}$ and $\rho = \frac{m}{k}$.

5.1 Copying

We can linearly combine two designs G_1 and G_2 by copying, which means that the new graph is defined to contain both G_1 and G_2 as separate components.

Proposition 4 (Copying). *If there exists a (k_1, m_1, t, E_1)-design G_1 and a (k_2, m_2, t, E_2)-design G_2 then there exists a $(k_1 + k_2, m_1 + m_2, t, E_1 + E_2)$-design to be denoted by $G_1 + G_2$.*

When two designs are linearly combined, the ε and ρ of the resulting graph is the linear combination of the two. (Let $\varepsilon(G)$ and $\rho(G)$ be the wiring complexity and redundancy of G).

$$\varepsilon(G_1 + G_2) = \frac{k_1}{k_1 + k_2}\varepsilon(G_1) + \frac{k_2}{k_1 + k_2}\varepsilon(G_2)$$
$$\rho(G_1 + G_2) = \frac{k_1}{k_1 + k_2}\rho(G_1) + \frac{k_2}{k_1 + k_2}\rho(G_2)$$

If G_1 is copied with itself (we can use the notation $G_1 + G_1 = 2G_1$), then the parameters of G doubles for k, m, and E. Notice that $\varepsilon(G_1) = \varepsilon(2G_1)$ and $\rho(G_1) = \rho(2G_1)$.

We can now use copying to show propositions about $\mathcal{R}_t$.

5.2 Region $\mathcal{R}_t$

Recall

Definition 2. *For a fixed $\mathcal{X}$ and $t \geq 1$ we define the region $\mathcal{R}_t$ as the closure of the set of all achievable pairs of (ε, ρ):*

$$\mathcal{R}_t \triangleq \text{closure}\left\{(\frac{E}{k}, \frac{m}{k}) : \exists(k, m, t, E) - \textit{design}\right\}$$

Proposition 5. *$(\varepsilon, \rho) \in \mathcal{R}_t$ if and only if there exists a sequence of (k, m, t, E)-designs with $\frac{m}{k} \to \rho$, $\frac{E}{k} \to \varepsilon$ and $k, m, E \to \infty$.*

Proof. From the definition of closure, $(\varepsilon, \rho) \in \mathcal{R}_t$ if and only if there is a sequence of points $\{(\varepsilon_i, \rho_i)\}_i \in \mathcal{R}_t$ approaching (ε, ρ). Each (ε_i, ρ_i) must be associated with some design G_i that is a (k_i, m_i, t, E_i)-design, where $m_i = \rho_i k_i$ and $E_i = \varepsilon_i k_i$. To show that $k, m, E \to \infty$, we can copy G_i with itself n_i times, where n_i is chosen so that $n_i k_i, n_i m_i, n_i E_i \to \infty$. □

Proposition 6. *If $(\varepsilon, \rho) \in \mathcal{R}_t$ and $\varepsilon' \geq \varepsilon$, $\rho' \geq \rho$, then $(\varepsilon', \rho') \in \mathcal{R}_t$.*

Proof. For each (ε, ρ), there is a $(k, \rho k, t, \varepsilon k)$-design G. We can create a sequence of G_i which are copies of G_i with an addition of some appropriate number of extra edges and extra redundant nodes added. Adding more edges and nodes does not decrease t. The sequence $(\varepsilon(G_i), \rho(G_i))$ can be created to converge to (ε', ρ'). □

Proposition 7. *$\mathcal{R}_t$ are closed convex subsets of $\mathbb{R}_+^2$.*

Proof. We need to show that if a pair of values (ε_1, ρ_1) and (ε_2, ρ_2) is in $\mathcal{R}_t$, then any point

$$(\varepsilon, \rho) = (\alpha\varepsilon_1 + (1-\alpha)\varepsilon_2, \alpha\rho_1 + (1-\alpha)\rho_2)$$

is also in $\mathcal{R}_t$ for $0 \leq \alpha \leq 1$.

There is are sequences $(\varepsilon_{1,i}, \rho_{1,i}) \to (\varepsilon_1, \rho_1)$ and $(\varepsilon_{2,i}, \rho_{2,i}) \to (\varepsilon_2, \rho_2)$, where for each i there exists a $(k_{1,i}, \rho_{1,i}k_{1,i}, t, \varepsilon_{1,i}k_{1,i})$-design $G_{1,i}$ and a $(k_{2,i}, \rho_{2,i}k_{2,i}, t, \varepsilon_{2,i}k_{2,i})$-design $G_{2,i}$.

We can find a sequence of rational numbers $\alpha_i = \frac{p_i}{q_i}$ where $p_i, q_i \in \mathbb{Z}_+$ and $\alpha_i \to \alpha$. Define integers $w_i = k_{1,i}(q_i - p_i)$ and $z_i = k_{2,i}p_i$. The copy $z_i G_{1,i} + w_i G_{2,i}$ achieves the point $(\varepsilon_i, \rho_i) = (\alpha_i\varepsilon_{1,i} + (1-\alpha_i)\varepsilon_{2,i}, \alpha_i\rho_{1,i} + (1-\alpha_i)\rho_{2,i})$ in $\mathcal{R}_t$ and $(\varepsilon_i, \rho_i) \to (\varepsilon, \rho)$. □

The regions $\mathcal{R}_t$ define what designs are asymptotically achievable. Given a pair (ε, ρ), if $(\varepsilon, \rho) \in \mathcal{R}_t$, then we know that there is some design whose redundancy and wiring complexity is within a small neighborhood of (ε, ρ). If (ε, ρ) is not in $\mathcal{R}_t$, then no design can be arbitrarily close to (ε, ρ). Defining $\mathcal{R}_t$ determines the fundamental limits of redundancy and wiring complexity on designs for error correcting value t.

5.3 Repetition Block and Complete Design Achievability

To determine what $\mathcal{R}_t$ is , we can easily see that there cannot exist a (k, m, t, E)-design where $\varepsilon < t$, so $\mathcal{R}_t$ cannot pass the vertical boundary $\varepsilon = t$.

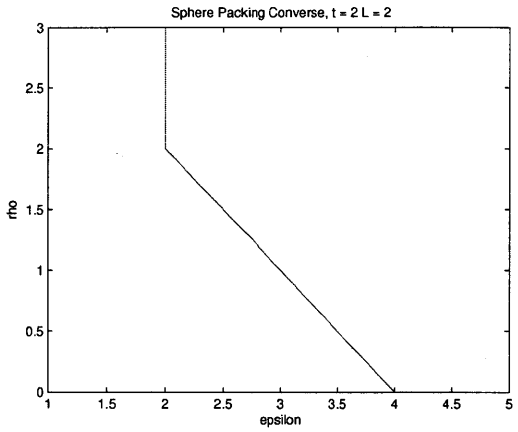


Figure 5-1: Plot of the boundary of $\mathcal{R}_t^{(K)}$ for $|\mathcal{X}| = 2$ and $t = 2$.

Next, recall that the repetition block $kK(1,t)$ can always achieve the point (t,t) and the complete design $K(k,t)$ can asymptotically converge to $(|\mathcal{X}|t, 0)$. By convexity, the line of points between (t,t) and $(|\mathcal{X}|t,0)$ must in $\mathcal{R}_t$.

Definition 10.

$$\mathcal{R}_t^{(K)} \triangleq \{(\varepsilon,\rho) : \varepsilon \geq |\mathcal{X}|t + (1-|\mathcal{X}|)\rho, \varepsilon \geq t, \rho \geq 0\}$$

Proposition 8 (Repetition Block and Complete Design Achievability)**.**

$$\mathcal{R}_t^{(K)} \subseteq \mathcal{R}_t$$

The corner points of $\mathcal{R}_t^{(K)}$ are achieved by the repetition blocks and complete designs.

$\mathcal{R}_t^{(K)}$ describes all the possible trade-offs between redundancy and wiring complexity for designs which only include the repetition block and complete design. For some cases this completely defines all of $\mathcal{R}_t$. Restricting to the case when $|\mathcal{X}| = 2$, $\mathcal{R}_1^{(K)} = \mathcal{R}_1$. More interestingly, $\mathcal{R}_2^{(K)} = \mathcal{R}_2$. (See Chapter 6). The boundary of region $\mathcal{R}_2^{(K)}$ is plotted in Figure 5-1 for $|\mathcal{X}| = 2$.

5.4 t Normalized Achievability Regions

If some $|\mathcal{X}|$, we plot all the regions $\mathcal{R}_t^{(K)}$ on the sample plot, we would see that each successive $\mathcal{R}_t^{(K)}$ region expands outwards as t increases. For studying the regions $\mathcal{R}_t$, what we are intuitively interested in is the shape of the region, since that shows what points beyond $\mathcal{R}_t^{(K)}$ are achievable. If we want to compare the shape of one region $\mathcal{R}_t$ to another $\mathcal{R}_\tau$, it is natural to scale the two regions so that the $\mathcal{R}_t^{(K)}$ and $\mathcal{R}_\tau^{(K)}$ match up. To do this, we can divide the coordinates of points in $\mathcal{R}_t$ by t, a process which we call normalizing. This way, all normalized regions will have the point $(1,1)$ and $(|\mathcal{X}|, 0)$. When we start to look at normalized regions, it makes sense to study

what the normalized regions converge to when t goes to infinity. This result will give the idea of what $\mathcal{R}_t$ tends to as t increases. We call this limiting region $\mathcal{R}_\infty$.

Definition 3.

$$\mathcal{R}_\infty \triangleq \text{closure}\{(\frac{m}{kt}, \frac{E}{kt}) : \exists(k, m, t, E) - \textit{design}\}$$

The notation $\frac{1}{t}\mathcal{R}_t$ will denote the achievable region when $\varepsilon = \frac{E}{k}$ and $\rho = \frac{m}{k}$ are each divided by t.

Proposition 9.

$$\mathcal{R}_\infty = \text{closure}\left\{\bigcup_{t=1}^{\infty} \frac{1}{t}\mathcal{R}_t\right\}$$

Proof. Any point $(\tilde{\varepsilon}, \tilde{\rho})$ in $\mathcal{R}_\infty$ and closure $\left\{\bigcup_{t=1}^{\infty} \frac{1}{t}\mathcal{R}_t\right\}$ must be realized with some sequence of $(k_i, \tilde{\rho}k_it_i, t_i, \tilde{\varepsilon}k_it_i)$-design. □

We refer to $\frac{m}{kt}$ as the normalized redundancy and $\frac{E}{kt}$ as the normalized wiring complexity.

5.5 Merging

Previously, we have shown convexity using copying. This section defines a different way of linearly combining designs called **merging**.

Proposition 10 (Merging). *If there exists a (k, m_1, t_1, E_1)-designG_1 and a (k, m_2, t_2, E_2)-design G_2 then there exists a $(k, m_1 + m_2, t_1 + t_2, E_1 + E_2)$-design denoted by $G_1 \vee G_2$.*

By merging, we mean for two designs both with k functional nodes to be joined together by identifying the functional nodes. To be specific, if we list the k functional nodes of design G_1 and G_2, by merging, we mean to identify the ith functional node of G_1 with the ith functional nodes of G_2. The resulting ith functional node of the merged combination $G_1 \vee G_2$ will have all its neighbors in G_1 and all its neighbors in G_2. The number of edges and redundant nodes adds. For any labeling $s^k \in \mathcal{X}^k$, we can always use the respective labelings r_1^m and r_2^m which G_1 and G_2 uses to correct a total of $t_1 + t_2$ errors.

(If two designs G_1 and G_2 have a different number of functional nodes, say k_1 and k_2 respectively, we can always merge k_1 copies of G_2 and k_2 copies of G_1.)

We can also always merge a design with itself. Doing this once just doubles all the parameters. Merging n times multiples all the parameters by n. Note that merging a design with itself, does decrease normalized wiring complexity or redundancy.

If we want to merge a set of graphs $\{G_i\}_i$, we can express that as $\vee_i G_i$.

Proposition 11. *$\mathcal{R}_\infty$ is a closed convex subset of $\mathbb{R}_+^2$.*

Proof. We need to show for any two points (ξ_1, ϱ_1) and (ξ_2, ϱ_2) be two points in $\mathcal{R}_\infty$, that

$$(\xi, \varrho) = (\alpha\xi_1 + (1-\alpha)\xi_2, \alpha\varrho_1 + (1-\alpha)\varrho_2)$$

is also in $\mathcal{R}_\infty$ for $0 \leq \alpha \leq 1$.

There is are sequences $(\xi_{1,i}, \varrho_{1,i}) \to (\xi_1, \varrho_1)$ and $(\xi_{2,i}, \varrho_{2,i}) \to (\xi_2, \varrho_2)$, where for each i, there is some error correcting value $t_{1,i}$ and $k_{1,i}$, where there is a design $G_{1,i}$ which is $(k_{1,i}, \varrho_{1,i}t_{1,i}k_{1,i}, t_{1,i}, \xi_{1,i}t_{1,i}k_{1,i})$-design and there is some error correcting value $t_{2,i}$ and $k_{2,i}$, where there is a design $G_{2,i}$ which is a $(k_{2,i}, \varrho_{2,i}t_{2,i}k_{2,i}, t_{2,i}, \xi_{2,i}t_{2,i}k_{2,i})$-design. $G_{1,i}$ has parameter $\varepsilon_{1,i} = \xi_{1,i}t_{1,i}$ and $\rho_{1,i} = \varrho_{1,i}t_{1,i}$ and $G_{2,i}$ has parameter $\varepsilon_{2,i} = \xi_{2,i}t_{2,i}$ and $\rho_{1,i} = \varrho_{1,i}t_{1,i}$.

For each i, we can merge $G_{1,i}$ with itself $t_{2,i}$ times to get a

$$(k_{1,i}, t_{2,i}\varrho_{1,i}t_{1,i}k_{1,i}, t_{2,i}t_{1,i}, t_{2,i}\xi_{1,i}t_{1,i}k_{1,i})\text{-design}$$

which we will call $G'_{1,i}$. We can also merge $G_{2,i}$ with itself $t_{1,i}$ times to get a

$$(k_{2,i}, t_{1,i}\varrho_{2,i}t_{2,i}k_{2,i}, t_{1,i}t_{2,i}, t_{1,i}\xi_{2,i}t_{2,i}k_{2,i})\text{-design}$$

which we will call $G'_{2,i}$. $G'_{1,i}$ and $G'_{2,i}$ correct the same number of errors, so we can now use copying (Proposition 7) that $(\varepsilon_i, \rho_i) = (\alpha t_{2,i}\varepsilon_{1,i} + (1-\alpha)t_{1,i}\varepsilon_{2,i}, \alpha t_{2,i}\rho_{1,i} + (1-\alpha)t_{1,i}\rho_{2,i})$ lies in $\mathcal{R}_{t_{1,i}t_{2,i}}$. When we normalize this, we get the value

$$\frac{\varepsilon_i}{t_{1,i}t_{2,i}} = \alpha\frac{\varepsilon_{1,i}}{t_{1,i}} + (1-\alpha)\frac{\varepsilon_{2,i}}{t_{2,i}} = \alpha\xi_{1,i} + (1-\alpha)\xi_{2,i}$$

Similarly, we can get

$$\frac{\rho_i}{t_{1,i}t_{2,i}} = \alpha\frac{\rho_{1,i}}{t_{1,i}} + (1-\alpha)\frac{\rho_{2,i}}{t_{2,i}} = \alpha\varrho_{1,i} + (1-\alpha)\varrho_{2,i}$$

The point $(\alpha\xi_{1,i} + (1-\alpha)\xi_{2,i}, \alpha\varrho_{1,i} + (1-\alpha)\varrho_{2,i})$ lies in $\mathcal{R}_\infty$ and this converges to (ξ, ϱ) as $i \to \infty$

□

Proposition 12. $\mathcal{R}_\infty = \limsup \frac{1}{t}\mathcal{R}_t$

Proof. By merging, we can always show that for any t, $\frac{1}{t}\mathcal{R}_t \subset \frac{1}{2t}\mathcal{R}_{2t} \subset \frac{1}{4t}\mathcal{R}_{4t} \subset \frac{1}{8t}\mathcal{R}_{8t} \ldots$. □

$\mathcal{R}_\infty$ represents the achievable region of designs when $t \to \infty$. A main goal of this thesis is to determine $\mathcal{R}_\infty$ which is done in Chapter 7.

Chapter 6

Fundamental Limits: Finite t

The goal of this section is to find lower bounds to $\mathcal{R}_t$. In particular it will determine that the optimal designs for $t = 1, 2$ and $|\mathcal{X}| = 2$ and for $t = 1$ and $|\mathcal{X}| = 3$ are a linear combination of the repetition block and the complete design.

6.1 Covering Converse

We have shown that $\mathcal{R}_t^{(K)} \subseteq \mathcal{R}_t$ in Section 5.3. In this section, we will start showing results for converse bounds to $\mathcal{R}_t$.

We focus our attention and assume that $|\mathcal{X}| = 2$ for this section.

The key to the converse result for the next few sections is the idea of covering. The main idea is to look at how many labels $s^k \in \mathcal{X}^k$ a given redundant labeling r^m "covers".

6.1.1 Covering Converse for $\mathcal{R}_1$

Definition 11. *Let $g_t(r^m)$ to be the set of labels $s^k \in \mathcal{X}^k$ for which the design G with label r^m and s^k is t error correcting. We will call $g_t(r^m)$ the cover of r^m.*

Each $g_t(r^m)$ can be understood as a k dimensional rectangle. For the design to be t error correcting, a single functional node u_i can only take on a value $a \in \mathcal{X}$ for which the neighborhood of u_i contains a at least t times. Given r^m, the possible values functional node u_i can take is independent of the possible values functional node u_j can take for $i \neq j$. To find to $g_t(r^m)$, it is sufficient to take the product over the possible values each functional node can take.

Theorem 3. *On alphabet size $|\mathcal{X}| = 2$, $\mathcal{R}_1$ must be contained in the region defined by $\epsilon \geq 1$ and $\varepsilon \geq 2 - \rho$.*

Proof. Let design G have m redundant nodes and k functional nodes. Let $\mathcal{X} = 0, 1$.

All labels $s^k \in \mathcal{X}^k$ need to be in $g_1(r^m)$ for some label r^m. So it must be that

$$|\bigcup_{r^m} g(r^m)| \geq |\mathcal{X}|^k$$

Then,

$$2^k = |\mathcal{X}|^k \leq |\bigcup_{r^m} g(r^m)| \leq \sum_{r^m} |g_1(r^m)| \leq 2^m \max_{r^m} |g_1(r^m)|$$

which implies that

$$2^{k-m} \leq \max_{r^m} |g(r^m)|$$

If $k \geq m$, for a single redundant node labeling to be able to cover s^{k-m} different labelings, at least $k-m$ functional nodes need to take on both values in $\mathcal{X}$. This implies that $k-m$ nodes need at least 2 edges.

If we count up all the edges, we get

$$\begin{aligned} E &\geq 2(k-m)+m \\ &\geq 2k-m \\ \varepsilon &\geq 2-\rho \end{aligned}$$

If $k < m$, we still need to satisfy the condition that each functional node needs at least 1 edge, which gives $\varepsilon \geq 1$.

□

The region matches $\mathcal{R}_1^{(K)}$ exactly.

6.1.2 Covering Converse for $\mathcal{R}_2$

In order to extend the covering converse to larger t, we use the following lemma.

Lemma 2. *In a t error correcting design G, two functional nodes which both have degree t cannot be connected to any of the same redundant nodes.*

Otherwise, if the two functional nodes were different, the two nodes cannot both have t neighbors with the same value as itself.

Theorem 4 (2 EC Converse). *On alphabet size $|\mathcal{X}| = 2$, $\mathcal{R}_2$ must be contained in the region defined by $\epsilon \geq 2$ and $\varepsilon \geq 4-\rho$.*

Proof. The proof of this follows exactly from the main idea of Theorem 3 using Lemma 2.

We also need that for a t error correcting design G, where l_t is the number of functional nodes of degree t, the number of redundant labelings $r^m \in \mathcal{X}^m$ where $g_t(r^m) \neq \emptyset$ is at most $|\mathcal{X}|^{m-l_t(t-1)}$. If r^m is such that the labeling of the t neighbors of each functional node of degree t are not all the same, then for this r^m, $g_t(r^m)$ is empty. The number of labelings r^m which have non-empty covers is then $|\mathcal{X}|^{m-l_t(t-1)}$,

since once a single neighbor of a functional node with degree t is determined, the other neighbors of that node can only have one value.

Only 2^{m-l_t} number of labelings r^m can be t correcting. Thus, there must be some labeling r^m which covers

$$\frac{2^k}{2^{m-l_t}} = 2^{k-m+l_t}$$

labelings s^k in $\mathcal{X}^k$. So at least $k-m+l_t$ nodes must be able to take both 0 and 1 as a value, meaning that these nodes must have at least 4 edges. l_t nodes must have 2 edges and $k-(k-m-l_t)-l_t = m-2l_t$ nodes must have degree 3 or more. So

$$\begin{aligned} E &\geq 4(k-m+l_t) + 2l_t + 3(m-2l_t) \\ &= 4k - m \\ \varepsilon &\geq 4 - \rho \end{aligned}$$

□

The region matches $\mathcal{R}_2^{(K)}$ exactly. With this theorem, the achievable region of 2 error correcting designs on alphabet size 2 is completely solved. In terms of redundancy and wiring complexity, a solution achieving the best trade-off for asymptotic k is one where only linear combinations of the repetition block and complete design are needed.

For other values of t, we can find an upper bound with $\mathcal{R}_t^{(K)}$ and a lower bound using the General Covering Bound.

6.1.3 Covering Converse for General Design

We now examine what bounds the covering converse creates for other error correcting values t and other alphabet sizes. This is a generalization of the same techniques used in the Theorem 4.

Definition 12. *l_i is the number of functional nodes in design G with degree i.*

Lemma 3. *For a t error correcting design G, where l_t is the number of functional nodes of degree t, the number of redundant labels $r^m \in \mathcal{X}^m$ where $g_t(r^m) \neq \emptyset$ is at most $|\mathcal{X}|^{m-l_t(t-1)}$*

In order for u_i to take on c different values under some r^m, it is necessary for u_i to have at least ct edges in the design, ie, $deg(u_i) \geq ct$. Using this, we can find an upper bound of the cardinality of $g_t(r^m)$,

$$|g_t(r^m)| \leq \prod_{u_i} \lfloor \frac{deg(u_i)}{t} \rfloor = 1^{l_t+l_{t+1}+\ldots+l_{2t-1}} 2^{l_{2t}+l_{2t+1}+\ldots+l_{3t-1}} 3^{l_{3t}+l_{3t+1}+\ldots+l_{4t-1}} \ldots$$

which holds for every r^m.

$\bigcup_{r^m} g_t(r^m)$ needs to cover all possible s^k in order for the design to be t correcting. So we need

$$|\bigcup_{r^m} g_t(r^m)| = |s^k| = |\mathcal{X}|^k$$

Using the union bound, we have

$$\begin{aligned}|\bigcup_{r^m} g_t(r^m)| &\leq \sum_{r^m} |g_t(r^m)| \\ &\leq |r^m| \prod_{s_i} \lfloor \frac{deg(s_i)}{t} \rfloor \\ &= |\mathcal{X}|^m 2^{l_{2t}+\ldots+l_{3t-1}} 3^{l_{3t}+\ldots+l_{4t-1}} \ldots\end{aligned}$$

So we must have that

$$|\mathcal{X}|^k \leq |\mathcal{X}|^m 2^{l_{2t}+\ldots+l_{3t-1}} 3^{l_{3t}+\ldots+l_{4t-1}} \ldots$$

in order for the design to be t correcting

$$|\mathcal{X}|^k \leq |\mathcal{X}|^{m-l_t(t-1)} 2^{l_{2t}+\ldots+l_{3t-1}} 3^{l_{3t}+\ldots+l_{4t-1}} \ldots \tag{6.1}$$

In addition to equation 6.1, we have the constraints

$$\sum_{i \geq t} l_i = k$$

$$\sum_{i \geq t} i l_i = E$$

$$l_i \geq 0$$

We can find the minimum E using integer linear programming. Since we are actually interested in minimum ε given ρ, we can divide all the equations by k and optimize the values of $\frac{l_i}{k}$.

This optimization would be used to find a converse bound for general t and $|\mathcal{X}|$. The result are boundary lines which look piecewise linear. For $t = 2$, it gives exactly the result in Theorem 4. Figure 6-1 gives a plot for when $t = 2$ and $|\mathcal{X}| = 3$.

This lower bound is not tight for other t and $|\mathcal{X}|$, but it is the best bound found for values of ε near $|\mathcal{X}|t$. It is also worth mentioning that the covering converse provides a linear bound at all finite values of t near the point $(|\mathcal{X}|t, 0)$ on the achievable region. This shows that if $\rho = 0$ it must be that $\varepsilon \geq |\mathcal{X}|t$.

There are other converse bounds we can find. For instance, we can determine a converse bound which labels the redundant nodes using majority vote of its neighbors, and use the average number of edges connecting nodes of the same value as an upper bound to the error correcting number. However, we will not discuss this since results in Chapter 7 can beat this converse. In Chapter 7, we can use asymptotic results to

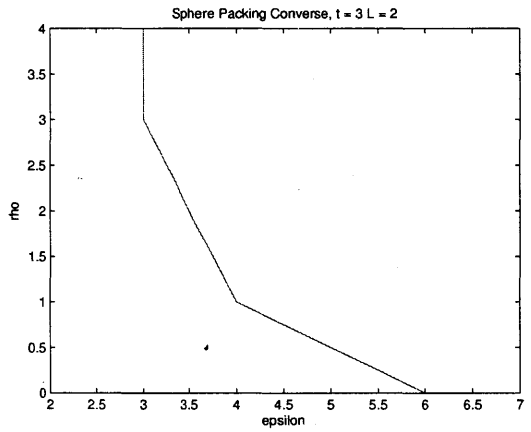


Figure 6-1: Plot of covering converse for the region $\mathcal{R}_2$ when $|\mathcal{X}| = 3$.

derive a lower bound which is better for values of ε away from $|\mathcal{X}|t$.

6.2 Ternary Alphabet and $t = 1$

While the covering converse can only be used as a lower bound for $|\mathcal{X}| = 3$, it turns out that on $|\mathcal{X}| = 3$, $\mathcal{R}_1$ is also equivalent to $\mathcal{R}_1^{(K)}$. To show this, a more sophisticated proof requiring details about graph structures is needed. The proof illustrates how tedious it possibly is to find tight converses for larger $|\mathcal{X}|$ and t.

Theorem 5. *For $|\mathcal{X}| = 3$ and $t = 1$ we have*

$$\mathcal{R}_1 = \{(\varepsilon, \rho) : \varepsilon \geq 3 - 2\rho, \rho \geq 0\}$$

and equivalent to $\mathcal{R}_t^{(K)}$.

Proof. Functional nodes can have labels in $\mathcal{X} = \{0, 1, 2\}$

Definition 13. *Define (*) to be the property that for any labeling of the functional nodes in $\mathcal{X}^k$, where k is the number of functional nodes, there exists a labeling of redundant nodes so that each functional node has at least one redundant node neighbor with the same labeling.*

The steps for this proof are:

1. Show that designs with functional nodes with degree 3 and greater can be disregarded.

2. Show that designs with a functional node of degree 1 cannot achieve better than $\mathcal{R}_1$.

3. Show that designs with functionals nodes all of degree 2 cannot achieve better than $\mathcal{R}_1$.

(a) Show designs containing two disjoint cycles connected by a path (see Figure 6-2(a)) violate (*)

(b) Show designs containing two cycles which intersect at one point (see Figure 6-2(b)) violate (*)

(c) Show designs containing two cycles which intersect at multiple points (see Figure 6-2(c)) violate (*)

Step (1)

For any (k, m, t, E)-design in $\mathcal{R}_1$, we can find an equivalent design where all functional nodes of degree 3 or more are in a separate component with an addition of finitely many more redundant nodes. Thus, it is sufficient to only prove for connected graphs with functional nodes with degree 1 and 2 only.

Step (2)

The key to this step is to show that if the design has a functional node of degree 1, the design must be a tree.

Definition 14. *We will call two functional nodes which share a redundant node* adjacent.

Consider if the design has a functional node u_0 of degree 1 and no functional nodes have degree 3 or more. Label u_0 a 0. In order to satisfy (*), the redundant node neighboring u_0 must also be labeled 0. Label functional nodes adjacent to u_0 the label 1. Since the degree of each functional node is at most 2, each of these functional nodes must have its remaining unlabeled redundant node labeled 1. We will call this labeling of functional nodes and neighboring redundant node the first iteration. On the nth iteration, we will label functional nodes adjacent to nodes labeled in the $n-1$ the iteration, alternating 0 and 1 (That means if iteration n labeled nodes 0, iteration $n+1$ will label the nodes 1 and vice versa.) The remaining neighboring redundant of each functional nodes labeled in a iteration must also be labeled in order to satisfy (*). Each iteration builds a tree of labeled nodes starting from u_0.

If there are any cycles, as the tree builds from u_0, we can label the first functional node to complete a cycle in the tree the label 2. The neighboring redundant nodes of this functional node are labeled either 0 or 1, so (*) is violated. Since the design is a tree, it must have the same at least the same number of redundant nodes as functional nodes, so the design lies in $\mathcal{R}_1$. We can now assume that all functional nodes are of degree 2.

Remark 1. *Notice having $|\mathcal{X}| \geq 3$ is important to avoid existence of even cycles.*

Step (3)

Our goal is to prove $\rho \geq \frac{1}{2}$. We will prove something stronger: For $k > 4$, we must have $m \geq k$ and for $k = 4$ we must have $m \geq 3$.

Recall that it is sufficient to prove this for designs on a single component. If there is a redundant node with degree 1, we can remove it with its neighboring functional node as a separate component, so all redundant nodes must also have degree 2 or more.

Definition 15. *We will call a labeling* alternating *if adjacent functional nodes have different labels.*

Remark 2. *If a design with k functional nodes, all of degree 2, and $k-1$ or fewer redundant nodes, all of degree 2 or 3, can be labeled alternatingly, then the design cannot satisfy (*).*

Proof. If there is an alternating labeling, at most each redundant node can only match the labeling of one of its neighboring functional nodes. There can only be at most $k-1$ matches, so there exists one functional node which does not have a neighboring redundant with the same label as itself. □

If the design does not have any cycles, then $\rho \geq \frac{1}{2}$ as in Step 2. So let A be a cycle in the design. In order for cycle A to not be a separate component, a redundant node with more than degree 2 must also be in A. Call this redundant node v_0. Call the neighbor of v_0 which is not in A u_0.

Let us build a path B starts at functional node u_0 as follows: The second node in path B will be the neighbor of u_0 which is not in cycle A. We can pick the next node in the path arbitrarily. The path ends when we reach a node in A or a node already in B. To show that (*) does not hold on the design, it is sufficient to show that (*) is not satisfied on $A \cup B$.

Depending on the endpoint of B, we have several cases:

Case (3a) **Endpoint of B coincides with an intermediate point of B.**

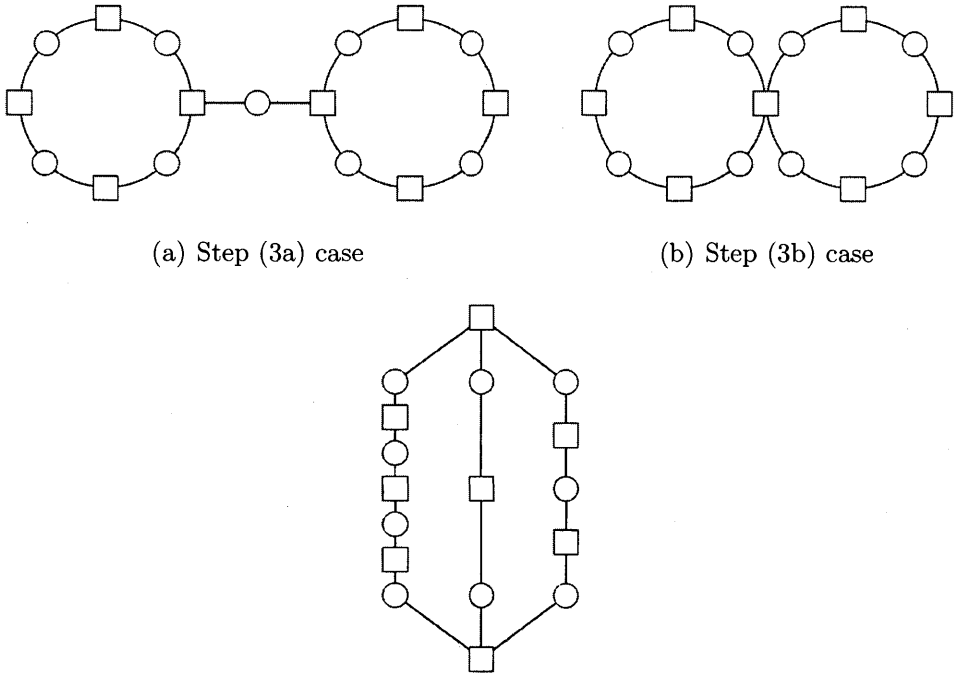
Let v_2 be the redundant node in path B where the path B ends. Rename the cycle created by path B to cycle C. The subgraph $A \cup B \cup C$ satisfies the conditions of Remark 2 so we need only show that we can find an alternating label. v_0 and v_2 are the two redundant node with degree 3. We can pick any alternatingly label for cycle A, then alternatingly label the remaining neighbor of v_0. Proceed to pick any alternating label for functional nodes in B. At v_2, we can label all three neighbors alternatingly, and still have a way to label the rest of cycle C alternatingly.

Case (3b) **Endpoint of B is node v_0.**

Let cycle C be the cycle formed using path B and v_0. As long as one of cycle A and cycle C have more than 2 functional nodes, the design violates (*).

Consider when the labeling is so that the two functional nodes in the larger cycle, assume this to be cycle A, neighboring v_0 are labeled the same value, say 0. The functional nodes in between are labeled alternatingly, which is possible because cycle A has at least three functional nodes. Let the two functional nodes neighboring v_0 in cycle C be labeled 1 and 2. v_0 must be labeled 0 in order for nodes in cycle A to each have a neighbor with the same label. Then if cycle C is labeled alternatingly, we will violate (*).

If both cycles have only 2 functional nodes, it is possible for this design to have $k = 4$ and $m = 3$ and satisfy (*). See Figure 6-3.



(a) Step (3a) case

(b) Step (3b) case

(c) Step (3c) case

Figure 6-2: Different cases for designs with degree 2 functional nodes

Case (3c) **Endpoint of B is some node of A different from v_0.**

Two redundant nodes in the design have degree at least 3. Call them v_0 and v_1. Cycle A and path B make up three distinct paths which go from v_0 to v_1, which we will refer to as E, F and G. As long as no two paths have only 1 functional node, then we can find an alternating label and use Remark 2.

Label $u_{i,X}$ to be the functional node neighboring v_i and in path X. If all paths E, F, and G have two or more functional nodes, label $0, 1, 2$ to $u_{0,E}, u_{0,F}, u_{0,G}$ and $1, 2, 0$ to $u_{1,E}, u_{1,F}, u_{1,G}$. Each path can be labeled alternatingly.

If there is one path with only one functional node, say path E, give the label $0, 1, 2$ to $u_{0,E}, u_{0,F}, u_{0,G}$ and $2, 1$ to $u_{1,F}, u_{1,G}$. Each path can be labeled alternatingly.

If two paths both have one functional node, say E and F, as long as the third path G has at least 3 functional nodes, the design can be labeled alternatingly. We can label the two functional nodes cycle created by paths E and F the values 0 and 1. Then since G has at least 3 functional nodes, we can label $u_{0,G}$ and $u_{1,G}$ the value 2 and label the rest of G alternatingly.

If G only has 2 functional nodes, then this is a design on $k = 4$ and $m = 3$ which satisfies (*). See Figure 6-4.

Note that the two exceptions with $k = 4$ are precisely the minimal non-trivial 1 error correcting designs discussed in Section 4.2.2.

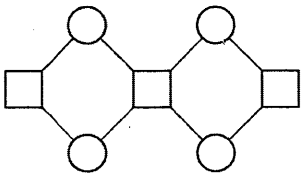


Figure 6-3: Design which satisfies (*). Exception to case (3b).

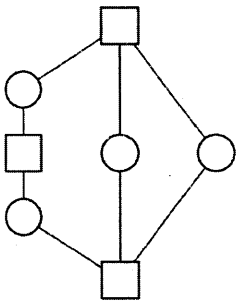


Figure 6-4: Design which satisfies (*). Exception to case (3c).

□

Chapter 7

Fundamental Limits: Asymptotic t

It is surprising that while no exact fundamental limit for the achievable region has been found for $t > 2$, the fundamental limit for when $t \to \infty$ can be determined.

For this entire section, we will assume that $|\mathcal{X}| = 2$, particularly that $\mathcal{X} = \{0, 1\}$. All the ideas are generalizable to larger alphabet sizes, but for the simplicity of the development, we choose to only focus on alphabet size of 2. Statements for larger alphabet will be discussed in Section 7.3.3.

Recall

Theorem 2 (Asymptotic t Result). *When $|\mathcal{X}| = 2$, the region $\mathcal{R}_\infty$ as defined in Definition 3 is the closure of points $(\tilde{\varepsilon}, \tilde{\rho})$ parameterized by P_S on $\mathbb{Z}_+$ with finite support and*

$$\tilde{\varepsilon} = \frac{\mathbb{E}[S]}{F(P_S)}, \tilde{\rho} = \frac{1}{F(P_S)},$$

$$F(P_S) \triangleq \min_{\lambda \in [0,1]} \max_{0 \le f_{l_0,l_1} \le 1} \min \left\{ \mathbb{E}\left[\frac{L_0}{\lambda} f(L_0, L_1)\right], \mathbb{E}\left[\frac{L_1}{1-\lambda}(1 - f(L_0, L_1))\right] \right\} \quad (3.1)$$

where the expectations are taken over $S \sim P_S$ and given S the distribution of $L_0 \sim Bino(S, \lambda)$ and $L_1 = S - L_0$.

7.1 Achievability: Subset Designs

The key proving to Theorem 2 is to use subset designs, which are key to both the achievability and the converse result. Subset designs are the provably optimal designs for asymptotic t. This turns out to be very lucky, because results for subset designs are easier to determine than for other designs due to their symmetry (as mentioned in Section 4.3.2).

Definition 16. *Let $T(n_0, n_1)$ be the type of a redundant node v in a design G where the functional nodes are labeled with some s^k. n_0 is the number of functional nodes of value 0 v is connected to, and n_1 is the number of functional nodes of value 1 v is connected to.*

7.1.1 Optimal Labeling of Redundant Nodes in Subset Designs

Consider a subset design $G = S(k, s)$ for some $s \geq 1$. Let $s^k \in \mathcal{X}^k = \{0, 1\}^k$ be a labeling of the functional nodes in G. Since we are only using two symbols, let λ be the proportion of functional nodes with label 0 and $1 - \lambda$ be the proportion of functional nodes with label 1[1]. The number of redundant nodes is$m = \binom{k}{s}$.

Given this labeling s^k of functional nodes, each redundant node has a type $T(l_0, l_1)$ where $l_0 + l_1 = s$. Let $P(l_0, l_1)$ be the proportion of redundant nodes with type $T(l_0, l_1)$.

$$\begin{aligned} P(l_0, l_1) &= \frac{\binom{k\lambda}{l_0}\binom{k(1-\lambda)}{l_1}}{\binom{k}{s}} \\ &= \frac{s!}{l_0! l_1!} \frac{((k\lambda)!/(k\lambda - l_0)!)((k(1-\lambda))!/(k(1-\lambda) - l_0)!)}{k!/(k-s)!} \\ &= \binom{s}{l_0} \lambda^{l_0} (1-\lambda)^{l_1} + o(1) \end{aligned}$$

For subset designs, the question of finding the optimal labeling of redundant nodes boils down to deciding how many redundant nodes of each type should be labeled a 0 and how many should be labeled a 1. Let $f(l_0, l_1)$ be the fraction of redundant nodes of type $T(l_0, l_1)$ which are labeled 0.

Proposition 13. *Subset design $S(k, s)$ can correct no more than*

$$t \leq \frac{1}{k}\binom{k}{s} \min_{\lambda \in I(k)} \max_{0 \leq f(l_0, l_1) \leq 1} \min \left\{ \mathbb{E}\left[\frac{L_0}{\lambda} f(L_0, L_1)\right], \mathbb{E}\left[\frac{L_1}{1-\lambda}(1 - f(L_0, L_1))\right] \right\}$$

error where $f(l_0, l_1)$ is represents the fraction of redundant node of type $T(l_0, l_1)$ labeled to 0 and the expectation is taken with P_{L_0, L_1}, the proportion of each redundant node type $T(l_0, l_1)$. $I(k) \triangleq \{0, 1/k, 2/k, ..., 1\}$.

Proof. For any $\lambda \in I(k)$, pick s^k to have λ proportion of 0's. We can count how many functional nodes with value 0 gain an effective neighbor (recall Definition 4). This can be expressed as $l_0 f(l_0, l_1)$. Similarly, the number of functional nodes with value 1 which gain an effective neighbor from this labeling of $T(l_0, l_1)$ is $l_1(1 - f(l_0, l_1))$. We can sum up these gains over all the types to get the total number of effective neighbors functional nodes with value 0 have, and we can do the same for the functional nodes with value 1. We can then define an average efficient neighbors for the 0 functional nodes as

$$y_0(\lambda) = \frac{1}{k_0} \sum_{(l_0, l_1)} m_{l_0, l_1} l_0 f(l_0, l_1)$$

[1]Due to permutation invariance (see Section 4.3.2), only the proportion of symbols in s^k matters.

and for the 1 functional nodes as

$$y_1(\lambda) = \frac{1}{k_1} \sum_{(l_0,l_1)} m_{l_0,l_1} l_1 (1 - f(l_0, l_1))$$

where k_0 and k_1 are the numbers of functional nodes with labeling 0 and 1 respectively, and m_{l_0,l_1} is the number of redundant nodes with type $T(l_0, l_1)$.

We can rewrite the equations above as

$$y_0(\lambda) = \frac{m}{\lambda k} \sum_{(l_0,l_1)} P(l_0, l_1) l_0 f(l_0, l_1) = \frac{m}{k} \mathbb{E}\left[\frac{L_0}{\lambda} f(L_0, L_1)\right]$$

$$y_1(\lambda) = \frac{m}{(1-\lambda)k} \sum_{(l_0,l_1)} P(l_0, l_1) l_1 (1 - f(l_0, l_1)) = \frac{m}{k} \mathbb{E}\left[\frac{L_1}{1-\lambda}(1 - f(L_0, L_1))\right]$$

The expectation is taken with $P(l_0, l_1)$ treated as a probability measure on $T(l_0, l_1)$.

Since quantities y_0 and y_1 count on average how many effective neighbors functional nodes with label 0 and 1 have, the error correcting value must be bounded above by the smaller of these two. So

$$t(\lambda) \leq \min\{y_0(\lambda), y_1(\lambda)\}$$

Since $S(k, s)$ needs to correct errors for all s^k, we can pick $\lambda \in I(k)$ to be the proportion which corrects the least number of errors. So

$$t = \min_{\lambda \in I(k)} t(\lambda)$$

□

While this is an upper bound on t, it turns out however that there exists a way to label the redundant nodes so that every functional node can get every close to having the same number of effective neighbors. In fact, it converges to the calculated average when the number of error corrected goes to infinity. Hence we will call this choice of $f(l_0, l_1)$ the optimal labeling.

Characteristics of Optimal Labeling

We will give an overview on what the optimal labeling of redundant nodes for subset designs need to look like in order to achieve the upper bound.

In any optimal labeling, redundant node types with larger values of l_0 should naturally be labeled the value 0 over redundant node types with smaller values of l_0.

Suppose $S(k, s)$ with functional node labeling s^k has two redundant nodes v_1 and v_2, where v_1 is of type $T(l_0, l_1)$ and v_2 is of type $T(j_0, j_1)$, so that $j_0 < l_0$ and $j_1 > l_1$. If the v_1 were given the label 0 and v_2 were given the label 1, we can always swap the labeling of v_1 and v_2 to create a redundant labeling which strictly increases the gains in effective neighbors.

Using this insight it must be that in the optimal redundant labeling of $S(k,s)$ for some functional labeling s^k, there exists a value j_0 and j_1 where $j_0 + j_1 = s$ such that

$$r_i = \begin{cases} 1 & \text{if the type } T(l_0,l_1) \text{ of } v_i \text{ is so that } l_1 > j_1 \\ 0 & \text{if the type } T(l_0,l_1) \text{ of } v_i \text{ is so that } l_0 > j_0 \\ 0 \text{ or } 1 & \text{if the type } T(l_0,l_1) \text{ of } v_i \text{ is so that } l_0 = j_0 \end{cases}$$

We can all the type $T(j_0,j_1)$ the splitting type, because the labeling of redundant nodes of this type is possibly split between 0's and 1's. All types with greater l_0 are labeled 0 and all with greater l_1 is labeled 1.

Optimal Labeling for a Merged Subset Design

While Proposition 13 is written for $S(k,s)$, it is not difficult to apply the same proof for $\vee_i S(k,s_i)$. $f(l_0,l_1)$ will be defined on each type for each subset size s_i. The expectation also needs to be adjusted slightly. Recall that for any merging $\vee_i S(k,s_i)$, we defined the weight $P_S(s)$ of each subset size (see Section 4.3.1). The expectation for multiple subsets will need to be over $P_S(s)P(l_0,l_1)$.

Proposition 14. *Subset design $\vee_i S(k,s_i)$ can correct no more than*

$$t \leq \frac{1}{k}\left(\sum_i \binom{k}{s_i}\right) \min_{\lambda \in I(k)} \max_{0 \leq f(l_0,l_1) \leq 1} \min\left\{\mathbb{E}\left[\frac{L_0}{\lambda} f(L_0,L_1)\right], \mathbb{E}\left[\frac{L_1}{1-\lambda}(1-f(L_0,L_1))\right]\right\}$$

error where $f(l_0,l_1)$ is represents the fraction of redundant node of type $T(l_0,l_1)$ labeled 0 and the expectation is taken with $P_S P_{L_0,L_1}$, the proportion of each redundant node with degree s type $T(l_0,l_1)$. $I(k) \triangleq \{0, 1/k, 2/k, ..., 1\}$

When we look at the optimal labeling $f(l_0,l_1)$ for a given s^k, we see that there is some ratio γ such that

$$f(l_0,l_1) = \begin{cases} 1 & \text{if } \frac{l_1}{l_0+l_1} > 1-\gamma \\ 0 & \text{if } \frac{l_0}{l_0+l_1)} > \gamma \\ f_\gamma & \text{if } \frac{l_0}{l_0+l_1} = \gamma. \end{cases}$$

This is shown in detail in Appendix B. We let the type $T(l_0,l_1)$ where $\frac{l_0}{l_0+l_1} = \gamma$ be called the splitting type and f_γ the splitting type ratio.

For a given subset design of weights P_S, the values $f(l_0,l_1)$ which attain the maximum in

$$\max_{0 \leq f(l_0,l_1) \leq 1} \min\left\{\mathbb{E}\left[\frac{L_0}{\lambda} f(L_0,L_1)\right], \mathbb{E}\left[\frac{L_1}{1-\lambda}(1-f(L_0,L_1))\right]\right\}$$

must also be for when

$$\mathbb{E}\left[\frac{L_0}{\lambda}f(L_0,L_1)\right] = \mathbb{E}\left[\frac{L_1}{1-\lambda}(1-f(L_0,L_1))\right]$$

Otherwise, we can always change $f(l_0,l_1)$ to increase the smaller value. Using this and the above form $f(l_0,l_1)$ must take, we can compute $f(l_0,l_1)$ exactly.

Proposition 15. *Let*

$$f'(l_0,l_1) = \frac{\mathbb{E}\left[\frac{L_0}{\lambda}\mathbb{I}\{\frac{L_0}{L_0+L_1} > \frac{l_0}{l_0,l_1}\}\right] - \mathbb{E}\left[\frac{L_1}{1-\lambda}\mathbb{I}\{\frac{L_0}{L_0+L_1} \leq \frac{l_0}{l_0,l_1}\}\right]}{\mathbb{E}\left[\frac{L_0}{\lambda}\mathbb{I}\{\frac{L_0}{L_0+L_1} = \frac{l_0}{l_0,l_1}\}\right] + \mathbb{E}\left[\frac{L_1}{1-\lambda}\mathbb{I}\{\frac{L_0}{L_0+L_1} = \frac{l_0}{l_0,l_1}\}\right]}$$

then

$$f(l_0,l_1) = \begin{cases} 1 & \text{if } f'(l_0,l_1) > 1 \\ 0 & \text{if } f'(l_0,l_1) < 0 \\ f'(l_0,l_1) & \text{if } 0 < f'(l_0,l_1) < 1. \end{cases}$$

where $f(l_0,l_1) = \arg\max_{0\leq f(l_0,l_1)\leq 1} \min\left\{\mathbb{E}\left[\frac{L_0}{\lambda}f(L_0,L_1)\right], \mathbb{E}\left[\frac{L_1}{1-\lambda}(1-f(L_0,L_1))\right]\right\}$. *Note that these functions are continuous and rational.*

7.1.2 Asymptotic Achievability

For a type $T(l_0,l_1)$ where $f(l_0,l_1)$ is either 0 or 1, all redundant nodes of the type are labeled the same label. If all $f(l_0,l_1)$ were either 0 or 1, due to the symmetry of subset designs, all functional nodes with label 0 will have an equal number of effective neighbors and the same holds for functional nodes with label 1. Thus, the average number of effective neighbors for each label equals the minimum number of effective neighbors for any functional node of that value.

The only reason the expression in Proposition 14 cannot hold for all k is because there is not necessarily a way to ensure that redundant nodes of the splitting type can be labeled in a way so that all functional nodes of a given label gain the same number of effective neighbors. We now present a designs which avoid this problem. This will show that the upper bound in Proposition 14 is asymptotically achievable. This is defined and shown below.

Proposition 16. *Given a set of rational weights P_S there exists a sequence of subset designs G_j*

- *Each design G_j has k_j functional nodes and m_j redundant nodes*
- *The weights of subsets in G_j are given by P_S*
- *Each design G_j can correct t_j errors*

so that

$$\frac{t_j}{\frac{m_j}{k_j}} \to \min_{\lambda\in[0,1]} \max_{0\le f(l_0,l_1)\le 1} \min\left\{\mathbb{E}\left[\frac{L_0}{\lambda}f(L_0,L_1)\right], \mathbb{E}\left[\frac{L_1}{1-\lambda}(1-f(L_0,L_1))\right]\right\} = F(P_S)$$

where expectation is taken with $P_S P_{L_0,L_1}$, where $P_{L_0,L_1} \sim Bino(S,\lambda)$.

Remark 3. *Because it will be unclear at times which distribution P_{L_0,L_1} we are taking the expectation to, we will sometimes use the notation $\mathbb{E}_{P_{L_0,L_1}}$ to specify $P_S P_{L_0,L_1}$. Since P_S is fixed, we do not need to specify that.*

Proof. Let

$$P_{k,\lambda}(l_0,l_1) = \frac{\binom{k\lambda}{l_0}\binom{k(1-\lambda)}{l_1}}{\binom{k}{s}}$$

and

$$P_\lambda(l_0,l_1) = \binom{s}{l_0}\lambda^{l_0}(1-\lambda)^{l_1}$$

We have that $P_{k,\lambda}(l_0,l_1) \to P_\lambda(l_0,l_1)$ as $k\to\infty$.
Let

$$h(k,\lambda) \triangleq \max_{0\le f(l_0,l_1)\le 1} \min\left\{\mathbb{E}_{P_{k,\lambda}}\left[\frac{L_0}{\lambda}f(L_0,L_1)\right], \mathbb{E}_{P_{k,\lambda}}\left[\frac{L_1}{1-\lambda}(1-f(L_0,L_1))\right]\right\}$$

and

$$h(\lambda) \triangleq \max_{0\le f(l_0,l_1)\le 1} \min\left\{\mathbb{E}_{P_\lambda}\left[\frac{L_0}{\lambda}f(L_0,L_1)\right], \mathbb{E}_{P_\lambda}\left[\frac{L_1}{1-\lambda}(1-f(L_0,L_1))\right]\right\}$$

We also have that $h(k,\lambda) \to h(\lambda)$ as $k\to\infty$ point-wise on λ. This is because both the function

$$\max_{0\le f(l_0,l_1)\le 1} \min\left\{\mathbb{E}_P\left[\frac{L_0}{\lambda}f(L_0,L_1)\right], \mathbb{E}_P\left[\frac{L_1}{1-\lambda}(1-f(L_0,L_1))\right]\right\}$$

is continuous over the probability distribution P.

We will design the sequence of subset designs G_j the following way. We will pick a sequence of increasing integers where $k_i \to \infty$.

For each s in the support of P_S, let $m_s = \binom{k_i}{s}$. Since each P_S is rational, we can express P_S with common denominator b. Express each $P_S(s)$ as the fraction $P_S(s) = \frac{a_s}{b\prod_s m_s}$. Let design G_i' to be a merging with $\frac{a_s}{m_s}$ merges of $S(k_i,s)$. Then G_i' has weights P_S.

Since λ represents the proportion of 0's, for each G_j, λ can take finitely many values in $I(k_j)$. (Recall $I(k) \triangleq \{0,1/k,2/k,...,1\}$). For each $\lambda \in I(k_j)$, we can

determine the optimal functions $f_{k_j,\lambda}(l_0, l_1)$ which attain $h(k_j, \lambda)$. Note from Proposition 15, all values of $f_{k_j,\lambda}(l_0, l_1)$ are all rational for each rational λ. Let q_j be a common denominator of all functions $f_{k_j,\lambda}(l_0, l_1)$ for each type and each λ.

Now let G_i be q_i merges of G_i'. G_i must have the same weights as G_i'. For any $\lambda \in I(K_i)$, there is a way to label the redundant nodes in G_i according to $f_{k_i,\lambda}(l_0, l_i)$ evenly. Then for any fractional values of $f_{k_i,\lambda}(l_0, l_1)$, we will label the redundant nodes of type $T(l_0, l_1)$ in the first $q_i f_{k_i,\lambda}(l_0, l_1)$ merges of G_i' in G_i the value 0. In the remaining $q_i(1 - f_{k_i,\lambda}(l_0, l_1))$ merges, redundant nodes of type $T(l_0, l_1)$ will be labeled 1. This is possible because we are guaranteed that $q_i(1 - f_{k_1,\lambda}(l_0, l_1))$ is an integer value.

Each functional node gets the same number of efficient neighbors from redundant nodes of type $T(l_0, l_1)$ under this labeling. Thus, the means that G_i with functional labeling proportion of λ, is able to correct precisely $\frac{m_1}{k_i} h(k_i, \lambda)$ errors.

Let

$$\lambda_i = \arg\min_{\lambda \in I(k_i)} h(k_i, \lambda)$$

. Then G_i corrects exactly $t_i = \frac{m_i}{k_i} h(k_i, \lambda_i)$ errors.

Since λ_j is on a compact set, a convergent subsequence must exist. Some subsequence λ_{j_a} must approach a limit $\hat{\lambda}$. Since $h(k_{j_a}, \lambda_{j_a})$ is also on a compact set[2], there must be a subsequence of designs G_{j_a}, call them G_{j_a}, where both $\lambda_{j_n} \to \hat{\lambda}$ and $h(k_{j_n}, \lambda_{j_n})$ converges.

Because $h(k, \lambda)$ converges to $h(\lambda)$ uniformly by Lemma 4 in Appendix C, it must be that $h(k_{j_n}, \lambda_{j_n}) \to h(\hat{\lambda})$. Then there exists a sequence of G_{i_n} which corrects $\frac{m_{i_n}}{k_{i_n}} h(\hat{\lambda})$. Since $h(\hat{\lambda}) \geq \min_{\lambda \in [0,1]} h(\lambda)$, that means the sequence G_{i_n} can also correct at least $\frac{m_{i_n}}{k_{i_n}} \min_{\lambda \in [0,1]} h(\hat{\lambda})$

□

7.1.3 Subset Achievable Region $\mathcal{R}_\infty^{(S)}$

From Proposition 16, there exists a subset design G_j where

$$\frac{\frac{t_j}{m_j}}{k_j} \to F(P_S)$$

. We have that $\rho(G_j) = \frac{m_j}{k_j}$ and $\varepsilon(G_j) = \frac{m_j \mathbb{E}[s]}{k_j}$

then the point

$$(\tilde{\varepsilon}, \tilde{\rho}) = (\frac{m_j \mathbb{E}[s]}{k_j t_j}, \frac{m_j}{k_j t_j}) = (\frac{\mathbb{E}[s]}{F(P_S)}, \frac{1}{F(P_S)})$$

must be in $\mathcal{R}_\infty$. The closure of such point will also be included.

We can define this region which is achievable by subset designs the region $\mathcal{R}_\infty^{(S)}$.

[2]Functions h are bounded

Definition 17.

$$\mathcal{R})\infty^{(S)} \triangleq \{(\tilde{\varepsilon}, \tilde{\rho}) : \exists P_S \text{ where } \tilde{\varepsilon} = \frac{\mathbb{E}[S]}{F(P_S)}, \tilde{\rho} = \frac{1}{F(P_S)}\}$$

where

$$F(P_S) \triangleq \min_{\lambda \in [0,1]} \max_{0 \leq f_{l_0,l_1} \leq 1} \min \left\{ \mathbb{E}\left[\frac{L_0}{\lambda} f(L_0, L_1)\right], \mathbb{E}\left[\frac{L_0}{1-\lambda}(1 - f(L_0, L_1))\right] \right\}$$

Currently we have shown that

$$\mathcal{R}_\infty^{(S)} \subseteq \mathcal{R}_\infty$$

7.2 Converse: Symmetrization

The surprising feature of subset designs is that they are guaranteed to perform better (or at least the same) as any other design. This is the converse result, which concludes that $\mathcal{R}_\infty^{(S)} = \mathcal{R}_\infty$, completing the proof of Theorem 2.

To prove this, we need the following lemma:

Proposition 17. *If there exists a (k, m, t, E)-design then there exists a symmetric $(k, mk!, tk!, Ek!)$-design.*

Proof. Let G be a (k, m, t, E)-design.

We will merge G with each of the $k!$ permutations of the functional nodes in G. G acts on a specific ordering of the k functional nodes. Different permutations G' do not change the graph structure, but will affect the merged combination $G \vee G'$.

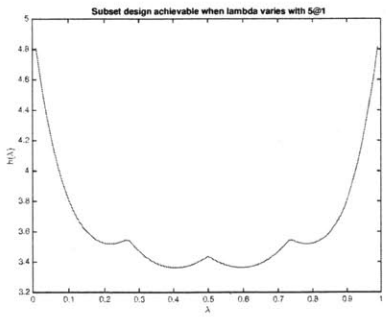
Let G_{PERM} be the result of merging all permutation. For any redundant node v in G, if v has degree s, every set of s nodes in G_{PERM} needs to be connected together by a copy of v. Thus, G_{PERM} is a $(k, mk!, tk!, Ek!)$-design and a subset design. □

Corollary 1 (Subset Bound Converse). *The achievable region $\mathcal{R}_\infty$ is contained the region $\mathcal{R}_\infty^{(S)}$.*

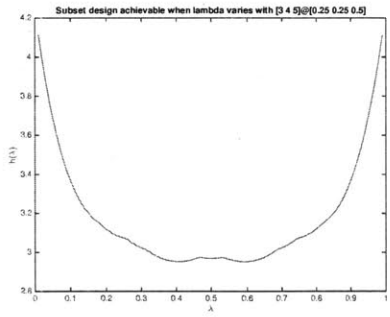
Proof. For any (k, m, t, E)-design in $\mathcal{R}_t$, we know there is a subset design which is a $(k, mk!, tk!, Ek!)$-design which achieves the same value of $(\frac{E}{kt}, \frac{m}{kt})$. □

While the achievable result for subset designs is for asymptotic t, even for finite t, we can apply Theorem 17 to get a converse result.

Remark 4. *Since $\frac{1}{t}\mathcal{R}_t \subset \mathcal{R}_\infty$, we can use the boundary defined in Theorem 2 can also be used as a converse for $\mathcal{R}_t$.*



(a) Single subset size example



(b) Merged subset example

Figure 7-1: Plots of $h(\lambda)$. Figure 7-1(a) is for P_S which has weight (1) on subset size (5). Figure 7-1(b) is for P_S which has weight $(.25, .25, .5)$ on $(3, 4, 5)$.

7.3 Discussion of Asymptotic Result

7.3.1 Evaluating $\mathcal{R}_\infty$

The optimization presented in Theorem 2 is difficult to evaluate exactly. The difficultly comes in that

- The optimization over λ is non-convex
- It is unknown which weights P_S achieve the boundary of $\mathcal{R}_\infty$

For a given P_S and λ, it is easy to evaluate

$$h(\lambda) \triangleq \max_{0 \le f(l_0,l_1) \le 1} \min \left\{ \mathbb{E}_{P_\lambda} \left[\frac{L_0}{\lambda} f(L_0, L_1) \right], \mathbb{E}_{P_\lambda} \left[\frac{L_1}{1-\lambda} (1 - f(L_0, L_1)) \right] \right\}$$

and hence we can find the plots of $h(\lambda)$ over λ.ïijĹExamples are plotted in Figure 7-1.

Finding the value of λ which attains the minimum for weights P_S when only a single subset is used is easier than when multiple subsets are used. For a single subset, it turns out interestingly that the minimum λ must be equivalent to the value f_γ of the splitting type ratio γ. However, results about single subset sizes are not helpful for finding $\mathcal{R}_\infty$ since the P_S which achieve the boundary of $\mathcal{R}_\infty$ seem to all require multiple subsets.

7.3.2 Numerical Approximation

Due to calculation difficulties, we will instead provide an approximation for the boundary. This approximation is a lower bound for $\mathcal{R}_\infty$. Thus, it still holds as a converse result for all $\mathcal{R}_t$. The lower bound is found using what we call the Max Probability method, which relaxes some of the optimization constraints so that linear programming can be used to find a solution. This method captures the linear structure of the within optimization problem. The derivation of this is given in Appendix D.

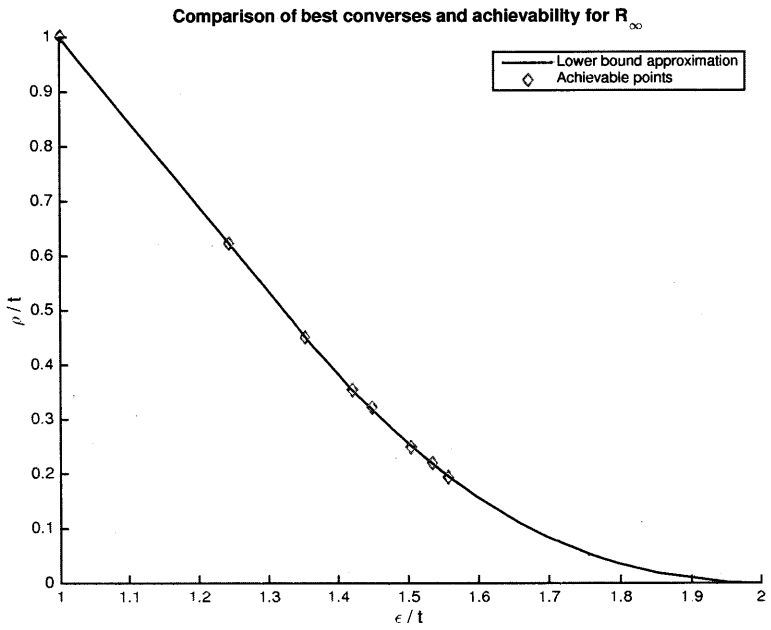


Figure 7-2: This figure plots points Max Probability converse bound and compares it with achievable points derived using the masses found in the converse bound. The achievable points used are arbitrarily selected.

The results are in Figure 7-2. As can be seen from the figure, the error in the approximation is mostly indistinguishable on the whole plot. It is on the order of 10^{-3}.

The numerical results give the following conjectures about what kinds of designs are optimal in terms of wiring complexity and redundancy.

- 4 or 5 subset sizes will make up most of the design
- Odd number subset sizes are favored
- The subset sizes which make up most of the design will largely be consecutive, possibly skipping even sizes

7.3.3 Larger Alphabet Size

When $|\mathcal{X}| > 2$, the converse result from the subset designs still applies, since Theorem 2 works regardless of alphabet size. We can define $\mathcal{R}_\infty^{(q)}$ for alphabet size q can still be defined in terms of what the subset designs can achieve.

Theorem 6 (Asymptotic t Result for Larger Alphabet). *When $|\mathcal{X}| = 2$, the region $\mathcal{R}_\infty$ as defined in Definition 3 is the closure of points $(\tilde{\varepsilon}, \tilde{\rho})$ parameterized by P_S on $\mathbb{Z}_+$ with finite support and where $\underline{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_q)$ where $\sum_{i=1}^{q} \lambda_i = 1$ and*

$$f(l_0, l_2, ..., l_q) = (f_1(l_0, l_2, ..., l_q), f_2(l_0, l_2, ..., l_q), ..., f_L(l_0, l_2, ..., l_q))$$

where $\sum_{i=1}^{q} f_i(l_0, l_2, ..., l_L) = 1$.

$$\tilde{\varepsilon} = \frac{[S]}{F(P_S)}, \tilde{\rho} = \frac{1}{F(P_S)},$$

$$F_L(P_S) = \min_{\lambda \in [0,1]} \max_{f(l_0,l_2,...,l_q)} \min_i \left\{ \mathbb{E}\left[\frac{L_i}{\lambda_i} f_i(L_0, L_2, ..., L_q) \right] \right\} \tag{7.1}$$

where the expectations are taken over $S \sim P_S$ *and the distribution of types* $T(l_0, ..., l_q)$ *is is given by proportions* $\underline{\lambda}$.

Chapter 8

Conclusion and Future Work

This thesis work developed a model for abstract physical redundancy and we are able to find the characteristics of optimal designs under given assumptions. Notably, we determined the fundamental trade-offs between wiring complexity and redundancy for small error correcting values on small alphabets and for asymptotic values error correcting values. We can also give the construction which leads to the fundamental results we found. The asymptotic result also provides a converse result for all designs. We can now provably determine that a design with certain number of edges and redundancy cannot achieve a certain number of errors.

There are some other questions we have not addressed. For instance, it would be ideal to find other small examples of optimal designs. Do to limited computing, this is difficult to do by simple search. It would be interesting to see if there were a class of designs which were optimal for a certain value of k. It would also be interesting to find the fundamental limits for other alphabet sizes and values of t larger than 2.

There is also some interest in solving the optimization in Theorem 2 exactly. It is possible that an exact solution may lead to some interesting finite designs.

The next direction we hope to take in our analysis of redundancy models is to see what is possible beyond the bipartite graph. For instance, what are the trade-offs when hierarchical models of redundancy are used. This model would include intermediate nodes which can facilitate connections of edges. The presence of the intermediate nodes can greatly reduce the number of edges. To correct t errors, we can connect each functional node to t intermediate nodes. Regardless of the number of functional nodes, the intermediate nodes can connect to finitely many redundant nodes. This way, we are able to achieve a wiring complexity of t and redundancy of 0 as the number of functional nodes goes to infinity. In such a case, we are interested in finding the fundamental tradeoffs with the number of intermediate nodes as a parameter.

Appendix A

NP-Hardness

If we are given a general design whose functional nodes are labeled, the problem of finding whether there is an label of redundant nodes which is t error correcting is NP-hard. This implies that if $P \neq NP$, then there is no general polynomial time algorithm that exists for labeling redundant nodes for all designs and all $s^k \in \mathcal{X}^k$.

To define our problem in the notation of complexity theory, we will define

$C = \{\langle G_{k,m}, s^k, t\rangle | G_{k,m}$ is a bipartite graph with k functional nodes and m redundant nodes, $s^k \in \mathcal{X}^k$ is a labeling of the functional nodes, so that there exists an labeling of the redundant nodes which is t error correcting$\}$

Theorem 7. *Computing C is NP-Hard.*

Proof. We will create a polynomial time reduction from $3SAT$ to our design problem C. For the reduction, we only need to consider when $t = 1$, since if the case of $t = 1$ alone is NP-hard, general t must also be NP-hard. Also, it is sufficient to assume $\mathcal{X} = \{0, 1\}$.

For any w which is a properly formatted boolean formula in conjunctive normal form with clauses exactly of three literals, we can define the reduction function to C as follows. There will be two redundant node for each variable x_i present in w. One redundant node will represent x_i and one will represent $\bar{x}_i$. Each x_i and $\bar{x}_i$ pair will both to connected to one functional node with label 0 and one functional node with label 1. This forces one redundant node in the pair to have a 0, representing a false, and one redundant node in the pair to have a 1 representing a true.

There will be one functional node with label 1 representing each clause. This functional node will be connected to the 3 literals which are present in the clause. This functional node can only be 1 error correcting if one of its neighbors is a 1, meaning one its literals are true.

If we can find a 1 error correcting redundant node labeling for this design, then we will also be able to satisfy w. If we can satisfy w, this leads to a 1 error correcting redundant node labeling. Thus, finding a 1 error correcting labeling in general is NP-hard.

□

Note however, that finding the redundant labeling for a specific class of designs is not necessarily NP-hard. We have already shown that for our subset designs, there is a very methodical way to find the optimal labeling which has polynomial complexity.

Also, the problem we are considering above applies to only one labeling s^k. To show that a design is t error correcting, we need to check every s^k, which is another level of exponential complexity.

Appendix B

Multiple Subsets Splitting Ratio

Even when G is a merged combination of multiple subset designs, there is the analogous idea of a splitting type ratio.

Proposition 18 (Splitting Type Ratio). *The optimal redundant labeling for a subset design G for some functional node labeling is described by the following. There exists a ratios γ where $0 \leq \gamma \leq 1$.*

$$r_i = \begin{cases} 1 & \text{if the type } T(l_0, l_1) \text{ of } v_i \text{ is so that } \frac{l_0}{l_0+l_1} > \gamma \\ 0 & \text{if the type } T(l_0, l_1) \text{ of } v_i \text{ is so that } \frac{l_0}{l_0+l_1)} < \gamma \\ 0 \text{ or } 1 & \text{if the type } T(l_0, l_1) \text{ of } v_i \text{ is so that } \frac{l_0}{l_0+l_1} = \gamma. \end{cases}$$

Proof. To show this, we need to only change the form of the optimization function. Instead of defining P_S where P_S is the proportion redundant nodes with degree s, we want to define $\hat{P}_S$ as the proportion of edges which are connected to redundant nodes of degree s.

Let E_s be that number of edges connected to a redundant node of degree s. Then

$$\hat{P}_S E = E_s = P_S m s$$

so we have the relations

$$\hat{P}_S \frac{1}{s} \varepsilon = P_S \rho$$

.

We want $\tilde{y}\rho = \hat{y}\varepsilon$, and it turns out the only change we need to make the optimization function for $\tilde{y}$ is to replace P_S with $\hat{P}_S \frac{1}{s}$.

$$\hat{y} = \max_{\hat{P}_S} \min_{0 \leq \lambda \leq 1} \max_{0 \leq f_{l_0,l_1} \leq 1} \min \left\{ \sum_{l_0+l_1=s} \hat{P}_S P_{l_0,l_1} \frac{l_0}{s\lambda} f_{l_0,l_1}, \sum_{l_0+l_1=s} \hat{P}_S P_{l_0,l_1} \frac{l_1}{s\bar{\lambda}} (1 - f_{l_0,l_1}) \right\} \tag{B.1}$$

The difference in this optimization function is that $\frac{l_0}{s}$ is used in place of l_0. Otherwise the form of the equation is exactly the same. So the labeling of redundant nodes must split instead along some ratio $\frac{l_0}{s}$ instead of some l_0 in the single subset case.

□

The main takeaway is that in the optimal labeling, there can be multiple types with all the same ratio, where some of the redundant nodes of these types are labeled 1 and the others are labeled 0. Redundant nodes of any other type are either all labeled to 0 or all labeled to 1.

Appendix C

Uniform Convergence for $h(k, \lambda)$

Lemma 4. *Let*

$$P_{k,\lambda}(l_0, l_1) = \frac{\binom{k\lambda}{l_0}\binom{k(1-\lambda)}{l_1}}{\binom{k}{s}}$$

and

$$P_{\lambda}(l_0, l_1) = \binom{s}{l_0}\lambda^{l_0}(1-\lambda)^{l_1}$$

Let

$$h(k, \lambda) \triangleq \max_{0 \le f(l_0,l_1) \le 1} \min\left\{\mathbb{E}_{P_{k,\lambda}}\left[\frac{L_0}{\lambda} f(L_0, L_1)\right], \mathbb{E}_{P_{k,\lambda}}\left[\frac{L_1}{1-\lambda}(1 - f(L_0, L_1))\right]\right\}$$

and

$$h(\lambda) \triangleq \max_{0 \le f(l_0,l_1) \le 1} \min\left\{\mathbb{E}_{P_{\lambda}}\left[\frac{L_0}{\lambda} f(L_0, L_1)\right], \mathbb{E}_{P_{\lambda}}\left[\frac{L_1}{1-\lambda}(1 - f(L_0, L_1))\right]\right\}$$

Then $h(k, \lambda)$ as $k \to \infty$ converges uniformly to $h(\lambda)$.

Proof. The maximizing set of functions $f^*(l_0, l_1)$ on types $T(l_0, l_1)$ on

$$\min\left\{\mathbb{E}_{P}\left[\frac{L_0}{\lambda} f(L_0, L_1)\right], \mathbb{E}_{P}\left[\frac{L_0}{1-\lambda}(1 - f(L_0, L_1))\right]\right\}$$

is continuous in P and λ by Proposition 15. Since the space of distributions P and λ are compact, the functions $f^*(l_0, l_1)$ must be uniformly continuous in P and λ.

We claim that $P_{j,\lambda}(l_0, l_1)$ converges uniformly to $P_{\lambda}(l_0, l_1)$ as $k \to \infty$.

$$
\begin{aligned}
P_{k,\lambda}(l_0,l_1) &= \frac{\binom{k\lambda}{l_0}\binom{k(1-\lambda)}{l_1}}{\binom{k}{s}} \\
&= \frac{s!}{l_0!l_1!}\frac{((k\lambda)!/(k\lambda-l_0)!)((k(1-\lambda))!/(k(1-\lambda)-l_0)!)}{k!/(k-s)!} \\
&= \binom{s}{l_0}\frac{(k\lambda)(k\lambda-1)...(k\lambda-l_0+1)(k(1-\lambda)(k(1-\lambda--1))...(k(1-\lambda)-l_1+1)}{(k)(k-1)...(k-s+1)} \\
&= \binom{s}{l_0}\left(\frac{k\lambda}{k}\right)\left(\frac{k\lambda-1}{k-1}\right).....\left(\frac{k(1-\lambda)-l_1+1}{k-s+1}\right) \\
&= \binom{s}{l_0}\left(\frac{\lambda}{1}\right)\left(\frac{\lambda-1/k}{1-1/k}\right).....\left(\frac{(1--\lambda)-(l_1-1)/k}{1-(s-1)/k}\right)
\end{aligned}
$$

We have that for small enough $\epsilon > 0$, there is some constant c

$$\left|\frac{\lambda-\epsilon}{1-\epsilon}-\lambda\right| = \epsilon\left|\frac{1-\lambda}{1-\epsilon}\right| = \epsilon C$$

So for $\epsilon > 0$, we can choose k so that $\frac{1}{k} < \frac{\epsilon}{2s^2c}$ so that

$$|P_{k,\lambda}(l_0,l_1) - P_\lambda(l_0,l_1)| \leq \left|\frac{\epsilon}{2s}(l_0\lambda + l_1(1-\lambda)) + o(\epsilon)\right| < \epsilon$$

for all λ.

Similarly, we can so that $\frac{l_0}{\lambda}P_{j,\lambda}(l_0,l_1)$ and $\frac{l_1}{1-\lambda}P_{j,\lambda}(l_0,l_1)$ also converge uniformly to $\frac{l_0}{\lambda}P_\lambda(l_0,l_1)$ and $\frac{l_1}{1-\lambda}P_\lambda(l_0,l_1)$ as $k \to \infty$.

Let $f^*_\lambda(l_0,l_1)$ be the optimal $f(l_0,l_1)$ for $h(\lambda)$ for each λ and let $f^*_{j,\lambda}(l_0,l_1)$ be the optimal $f(l_0,l_1)$ for $h(k_j,\lambda)$ for each λ.

Then $f^*_{j,\lambda}(l_0,l_1) \to f^*_\lambda(l_0,l_1)$ uniformly over λ. This is because since f that maximizes $h(k,\lambda)$ is uniformly continuous, for every ϵ, we can find δ such that if $|P_{j,\lambda}(l_0,l_1) - P_\lambda(l_0,l_1)| < \delta$, then $|f^*_{j,\lambda}(l_0,l_1) - f^*_\lambda(l_0,l_1)| < \epsilon$ for any λ. Then for every δ, we can find j large enough so that $|P_{j,\lambda}(l_0,l_1) -- P_\lambda(l_0,l_1)| < \delta$ for any λ, due to uniform convergence of $P_{j,\lambda}(l_0,l_1)$.

For any ϵ, there exists a k_j large enough that

$$
\begin{aligned}
|h(\lambda) - h(k_j, \lambda)| &= \left| h(\lambda) - \min \left\{ \sum_{l_0+l_1=s} P_S P_{j,\lambda}(l_0, l_1) \frac{L_0}{\lambda} f^*_{j,\lambda}(L_0, L_1), \right. \right. \\
&\quad \left. \left. \sum_{l_0+l_1=s} P_S P_{j,\lambda}(l_0, l_1) \frac{L_1}{1-\lambda} (1 - f^*_{j,\lambda}(L_0, L_1)) \right\} \right| \\
&= \left| h(\lambda) - \min \left\{ \sum_{l_0+l_1=s} P_S (P_\lambda(l_0, l_1) \frac{L_0}{\lambda} + \epsilon_P(l_0, l_1)) \right. \right. \\
&\quad (f^*_\lambda(L_0, L_1) + \epsilon_f(l_0, l_1)), \\
&\quad \left. \left. \sum_{l_0+l_1=s} P_S (P_\lambda(l_0, l_1) \frac{L_1}{1-\lambda} + \epsilon_P(l_0, l_1))(1 - (f^*_\lambda(L_0, L_1) + \epsilon_f(l_0, l_1))) \right\} \right| \\
&\leq c \sum_{l_0+l_1=s} |\epsilon_P(l_0, l_1)| + |\epsilon_f(l_0, l_1)| + |\epsilon_P(l_0, l_1)\epsilon_f(l_0, l_1)| \\
&< \epsilon
\end{aligned}
$$

for all λ.

□

Appendix D

Numerical Results Derivation

D.1 Approximation with lower bound

The optimization problem in Theorem 2 is difficult to solve exactly, but it can be bounded with an upper and lower bound which are very close.

We will approximate the best point $(\tilde{\varepsilon}, \tilde{\rho})$ achievable in $\mathcal{R}_\infty$ given any weights P_S with a fixed value $\mathbb{E}[S] = \varepsilon_r$.

Define

$$\tilde{y} = \max_{P_S:\mathbb{E}[S]=\varepsilon_r} F(S)$$

.

The notation $P_S(s)$ will specify the weight at s.

Starting with Equation (7.1), we can rewrite the optimization in terms of the splitting type ratio γ, where the optimal labeling will have

$$f_{l_0,l_1} = \begin{cases} 1 & \text{if } \frac{l_1}{l_0+l_1} > 1-\gamma \\ 0 & \text{if } \frac{l_0}{l_0+l_1)} > \gamma \\ f_\gamma & \text{if } \frac{l_0}{l_0+l_1} = \gamma. \end{cases}$$

which will give

$$\tilde{y} = \max_{P_S} \min_{0\le\lambda\le 1} \max_{\substack{0\le\gamma\le 1 \\ 0\le f_\gamma\le 1}} \left\{ \frac{1}{\lambda} \left(\mathbb{E}[l_0 \mathbb{I}[l_0 > s\gamma]] + \mathbb{E}[l_0 f_\gamma \mathbb{I}[l_0 = s\gamma]] \right), \right.$$
$$\left. \frac{1}{\bar{\lambda}} \left(\mathbb{E}[l_1 \mathbb{I}[l_0 < s\gamma]] + \mathbb{E}[l_1(1-f_\gamma)\mathbb{I}[l_0 = s\gamma]] \right) \right\}$$

Let $V_\lambda(n) \sim \text{Bino}(n, \lambda)$ that is V_λ is a binomial variable with probability of 0 equal to λ. $\bar{\lambda} = 1 - \lambda$.

$$\begin{aligned}
\frac{1}{\lambda}\mathbb{E}[l_0\mathbb{I}[l_0 > s\gamma]] &= \sum_{l_0=0}^{s}\frac{1}{\lambda}\text{Bino}(l_0,s,\lambda)l_0\mathbb{I}[l_0 > s\gamma] \\
&= \sum_{l_0=0}^{s}\frac{1}{\lambda}\binom{s}{l_0}\lambda^{l_0}(1-\lambda)^{s-l_0}l_0\mathbb{I}[l_0 > s\gamma] \\
&= \sum_{l_0=0}^{s}\frac{s!}{(l_0-1)!(s-l_0)!}\lambda^{l_0-1}(1-\lambda)^{s-l_0}\mathbb{I}[l_0 > s\gamma] \\
&= \sum_{l_0=0}^{s}s\binom{s-1}{l_0-1}\lambda^{l_0-1}(1-\lambda)^{(s-1)-(l_0-1)}\mathbb{I}[l_0 > s\gamma] \\
&= s\sum_{l_0=0}^{s}\text{Bino}(l_0-1,s-1,\lambda)\mathbb{I}[l_0 > s\gamma] \\
&= s\sum_{j=0}^{s-1}\text{Bino}(j,s-1,\lambda)\mathbb{I}[j > s\gamma-1] \\
&= s\mathbb{P}[V_\lambda(s-1) > s\gamma-1]
\end{aligned}$$

and

$$\frac{1}{\lambda}\mathbb{E}[l_0 f_\gamma\mathbb{I}[l_0 = s\gamma]] = sf_\gamma\mathbb{P}[V_\lambda(s-1) = s\gamma-1]$$

Similarly,

$$\begin{aligned}
\frac{1}{\lambda}\mathbb{E}[l_1\mathbb{I}[l_0 < s\gamma]] &= \sum_{l_0=0}^{s}\frac{1}{1-\lambda}\text{Bino}(l_0,s,\lambda)(s-l_0)\mathbb{I}[l_0 < s\gamma] \\
&= \sum_{l_0=0}^{s}\frac{1}{1-\lambda}\binom{s}{l_0}\lambda^{l_0}(1-\lambda)^{s-l_0}(s-l_0)\mathbb{I}[l_0 < s\gamma] \\
&= \sum_{l_0=0}^{s}s\binom{s-1}{l_0}\lambda^{l_0}(1-\lambda)^{s-1-l_0}\mathbb{I}[l_0 < s\gamma] \\
&= s\sum_{l_0=0}^{s}\text{Bino}(l_0,s-1,\lambda)\mathbb{I}[l_0 < s\gamma] \\
&= s\sum_{j=0}^{s}\text{Bino}(j,s-1,\lambda)\mathbb{I}[j < s\gamma] \\
&= s\mathbb{P}[V_\lambda(s-1) < s\gamma]
\end{aligned}$$

and

$$\frac{1}{\lambda}\mathbb{E}[l_1(1-f_\gamma)\mathbb{I}[l_0 = s\gamma]] = s(1-f_\gamma)\mathbb{P}[V_\lambda(s-1) = s\gamma]$$

We can simplify the quantities

$$\frac{1}{\lambda}\mathbb{E}_{s,\lambda}\left[l_0\mathbb{I}[l_0 > s\gamma] + l_0 f_\gamma \mathbb{I}[l_0 = s\gamma]\right] = \sum_s P_S(s)s(\mathbb{P}[V_\lambda(s) > s\gamma - 1] + f_\gamma \mathbb{P}[V_\lambda(s) = s\gamma - 1])$$

$$\frac{1}{\lambda}\mathbb{E}_{s,\lambda}\left[l_1\mathbb{I}[l_0 < s\gamma] + l_1(1-f_\gamma)\mathbb{I}[l_0 = s\gamma]\right] = \sum_s P_S(s)s(\mathbb{P}[V_\lambda(s) < s\gamma] + (1-f_\gamma)\mathbb{P}[V_\lambda(s) = s\gamma])$$

$$\tilde{y} = \max_{P_S} \min_{0\le\lambda\le 1} \max_{\substack{0\le\gamma\le 1 \\ 0\le f_\gamma\le 1}} \min\left\{\sum_s P_S(s)s(\mathbb{P}[V_\lambda(s) > s\gamma - 1] + f_\gamma\mathbb{P}[V_\lambda(s) = s\gamma - 1]),\right.$$
$$\left.\sum_s P_S(s)s(\mathbb{P}[V_\lambda(s) < s\gamma] + (1-f_\gamma)\mathbb{P}[V_\lambda(s) = s\gamma])\right\}$$

We can convexify the minimum

$$\tilde{y} = \max_{P_S} \min_{0\le\lambda\le 1} \max_{\substack{0\le\gamma\le 1 \\ 0\le f_\gamma\le 1}} \min_{0\le c\le 1}\left\{c\sum_s P_S(s)s(\mathbb{P}[V_\lambda(s) > s\gamma - 1] + f_\gamma\mathbb{P}[V_\lambda(s) = s\gamma - 1])\right.$$
$$\left. + (1-c)\sum_s P_S(s)s(\mathbb{P}[V_\lambda(s) < s\gamma] + (1-f_\gamma)\mathbb{P}[V_\lambda(s) = s\gamma])\right\}$$

When we set $c = \frac{1}{2}$, this creates an upper bound in which we can continue to simplify

$$\tilde{y} \le \max_{P_S} \min_{0\le\lambda\le 1} \max_{\substack{0\le\gamma\le 1 \\ 0\le f_\gamma\le 1}} \left\{\frac{1}{2}\sum_s P_S(s)s(1 + f_\gamma\mathbb{P}[V_\lambda = s\gamma - 1]) + (1-f_\gamma)\mathbb{P}[V_\lambda(s) = s\gamma])\right\} \tag{D.1}$$

$$\le \max_{P_S} \min_{0\le\lambda\le 1} \left\{\frac{1}{2}\sum_s P_S(s)s(1 + \max_{\substack{0\le\gamma\le 1 \\ 0\le f_\gamma\le 1}} (f_\gamma\mathbb{P}[V_\lambda = s\gamma - 1]) + (1-f_\gamma)\mathbb{P}[V_\lambda(s) = s\gamma]))\right\} \tag{D.2}$$

$$= \max_{P_S} \min_{0\le\lambda\le 1} \left\{\frac{1}{2}\sum_s P_S(s)s(1 + \max_{0\le x_s\le s-1}(\mathbb{P}[V_\lambda(s) = x_s]))\right\} \tag{D.3}$$

Equation D.3 is what we call the Max Probability Bound, since deriving it requires finding the value x_s which achieves the largest probability value under Binomial$(s-1, \lambda)$. We can define

$$\phi_s(\lambda) = \frac{s}{2}(1 + \max_{0 \le x_s \le s-1}(\mathbb{P}[V_\lambda(s) = x_s]))$$

so that

$$\tilde{y} \le \max_{P_S} \min_{0 \le \lambda \le 1} \sum_s P_S(s)\phi_s(\lambda)$$

We want the minimum value over λ, but since $\phi_s\lambda$ is not convex in λ, this is a difficultly in the the optimization. However, since we are only interested in an upper bound, we can relax the condition and choose the minimal λ over a countable set of λ. Let this set be

$$L = \{\lambda \in [0,1] : \phi_s(\lambda) \le \phi_s(\lambda') \text{ for some } s \text{ and } \forall \lambda' \in [0,1]\}$$

which is the set of all λ which minimizes ϕ_s for some s. The set of minimal λ is exactly the set

$$L = \left\{ \frac{\lfloor s/2 \rfloor}{s} \right\}_s$$

For any $L \subset [0,1]$, we must have that

$$\tilde{y} \le \max_{P_S} \min_{0 \le \lambda \le 1} \sum_s P_S(s)\phi_s(\lambda) \le \max_{P_S} \min_{\lambda \in L} \sum_s P_S(s)\phi_s(\lambda)$$

[1]

Since for each subset size s, there is one value of λ which is the minimizing value when s is the only subset, we can refer to this λ as $\lambda_{min}(s)$. We can equivalently write

$$\max_{P_S} \min_{\lambda \in L} \sum_s P_S(s)\phi_s(\lambda) = \max_{P_S} \min_i \sum_s P_S(s)\phi_s(\lambda_{min}(i)) = \max_{P_S} \min_i \sum_s P_S(s)\phi_{s,i}$$

where we define $\phi_{s,i} = \phi_s(\lambda_{min}(i))$.

Given a parameter $\varepsilon_r = \sum_s P_S(s)s$, solving for $\max_{P_s} \min_i \sum_s P_S(s)\phi_{s,i}$ is an infinite dimensional linear program.

For values where ε_r is finite, there are only a finite number of subset sizes have non-zero weight in the optimal labeling (see Appendix D.3). So practically, we can solve a finite dimensional linear program to find an upper bound for $\tilde{y}$ given each ε_r.

The linear program for n subsets is as follows:

[1]The specific L we choose is conjectured to make the inequality tight. We conjecture this because the plot for $\phi_s(\lambda)$ looks to be concave at most points, and not differentiable at points $\frac{x}{s}$ where x is an integer. The function $\sum_s P_S(s)\phi_s(\lambda)$ will also be concave at most points. Thus, the minimum is probably at a non-differentiable point.

$$\textbf{maximize } t \tag{D.4a}$$

$$\textbf{subject to } \sum_s P_S(s)\phi_{s,i} \geq t, \forall 0 \leq i \leq n \tag{D.4b}$$

$$P_S(s) \geq 0, \forall s \tag{D.4c}$$

$$\sum_{s=1}^{n} P_S(s) = 1 \tag{D.4d}$$

$$\sum_{s=1}^{n} P_S(s)s = \varepsilon_r \tag{D.4e}$$

We can run this linear program to plot points on the Max Probability converse bound.

D.2 Achievability Approximation

To show a point in R_∞ is achievable, it is sufficient to find a set of masses P_S for each the optimization equation solved for P_S achieves that point. Searching all possible masses P_S is not computationally efficient. It turns out we can get decently close to the lower bound approximation by using the same masses the lower bound approximation used. For each ε_r, the lower bounds finds masses P_s which are optimal for ε_r in the lower bound equation. We can use the same P_s and see what values are achieved using the original optimization equation.

The result for some arbitrary ε_r are also plotted in Figure 7-2. The calculation of these is shown in Table D.1

For each ε_r we look at the assignments of weights of subset sizes, that leads to the upper bound of $\tilde{y}$. ε_r can take on any positive real value greater than 1.

D.3 Finitely Many Subsets

To show what only finitely many subsets have positive weights when ε_r is positive, we start by taking the dual of the optimization in Equations (D.4a) to (D.4e).

The Lagrangian is

$$L(P_s, \pi_i, \nu_s, \eta, \mu) = t + \sum_i \pi_i \left(-\sum_s P_s(s)\phi_{s,i} + t \right) - \sum_s \nu_s P_s(s) + \eta \left(\sum_s P_s - 1 \right) + \mu \left(\sum_s P_s(s)s - \varepsilon_r \right)$$

so that the dual problem is

Converse bounds and achievable points table

ε_r	Converse point	Achievable point using converse
2	[1,3,4,5] [.6203,.2093,.1003,.0701] **1.6081** (1.2437,.6219)	 **1.6076** (1.2441,.6221)
3	[1,3,4,5] [.2405,.4187,.2006,.1402] **2.2162** (1.3537,.4512)	 **2.2152** (1.3543,.4514)
4	[3,4,5,6,7] [.5223,.2095,.1311,.0204,.1168] **2.81855** (1.4192,.4512)	 **2.817467** (1.4197,.3549)
5	[3,4,5,7] [.3133,.2347,.2774,.1746] **3.4060** (1.4680,.2936)	 **3.4058** (1.4681,.2936)
6	[5,6,7,9] [.4450,.3125,.1411,.1013] **3.9892** (1.5041,.2507)	 **3.9885** (1.5042,.2507)
7	[5,6,7,8,9] [.3505,.0033,.1342,.3197,.1923] **4.5648** (1.5335,.2191)	 **4.5644** (1.5336,.2191)
8	[7,8,9,11] [.3969,.3616,.1638,.0777] **5.1363** (1.5575,.1947)	 **5.1364** (1.5576,.1947)

Table D.1: This table compares the converse bound and arbitrarily selected achievable points for different values of ε_r. The first column titled "Converse point " is a converse bound obtained using the Max Probability optimization. The top entry in each box is the subsets used and the entry below this is the weights, respectively, of each subset used. The bolded entry is the value of $\tilde{y}$ and the entry below that is the point $(\frac{\varepsilon}{t}, \frac{\rho}{t})$. The second column titled "Achievable point using converse" uses the subsets and weights from the converse to see what $\tilde{y}$ and point would be achieved.

$$\textbf{maximize} \quad -t + \sum_i \pi_i t - \eta - \varepsilon_r \mu \tag{D.5a}$$

$$\textbf{subject to} \quad -\sum_i \pi_i \phi_{s,i} + \eta + \mu s \geq 0, \forall s \tag{D.5b}$$

$$\pi_i \geq 0, \forall i \tag{D.5c}$$

Complementary Slackness implies that

$$P_S(s)\left(-\sum_i \pi_i \phi_{s,i} + \eta + \mu s\right) = 0, \forall s$$

so for all s such that $P_S(s) = 0$, it must be that the inequality $-\sum_i \pi_i \phi_{s,i} + \eta + \mu s \geq 0$ holds with equality.

$$-\sum_i \pi_i \phi_{s,i} + \eta + \mu s \geq 0$$

$$-\sum_i \pi_i \frac{s}{2}(1 + \max_{0 \leq x_s \leq s-1}(\mathbb{P}[V_{\lambda_i}(s) = x_s])) + \eta + \mu s \geq 0$$

$$-\frac{1}{2}\sum_i \pi_i - \frac{1}{2}\sum_i \pi_i \max_{0 \leq x_s \leq s-1}(\mathbb{P}[V_{\lambda_i}(s) = x_s]) + \frac{\eta}{s} + \mu \geq 0$$

$$\left(\mu - \frac{1}{2}\sum_i \pi_i\right) + \frac{\eta}{s} \geq \frac{1}{2}\sum_i \pi_i \max_{0 \leq x_s \leq s-1}(\mathbb{P}[V_{\lambda_i}(s) = x_s])$$

The value of $\max_{0 \leq x_s \leq s-1}(\mathbb{P}[V_{\lambda_i}(s) = x_s])$ decreases monotonically to 0 on any λ_i as $s \to \infty$. So given any set of π_i, there is some s_0 where for every $s > s_0$, $\frac{1}{2}\sum_i \pi_i \max_{0 \leq x_s \leq s-1}(\mathbb{P}[V_{\lambda_i}(s) = x_s]) < \mu - \frac{1}{2}\sum_i \pi_i$. So this constraint cannot be tight for any $s > s_0$, and so $P_S(s) = 0$.

Appendix E

Characteristic of Minimzing λ

The following is a statement we can make about the λ used to minimize the expression for $F(S)$ when only one subset size is used. We will use the name notation used in sec::numresults.

Proposition 19. *For given s where $\hat{P}_S(s) = 1$ (only one subset is used), the values of λ which are local minimums are values of λ where $f_\gamma = \lambda$.*

Since piecewise portions of the function for $\hat{y}$ is convex, so each value of γ will have one local minimum. The function for s should have $s-1$ local minimums.

Proof. For when $\hat{P}_S = 1$ for a particular s, then

$$\hat{y} = \mathbb{P}[V_\lambda(s-1) > s\gamma - 1] + f_\gamma \mathbb{P}[V_\lambda(s-1) = s\gamma - 1]$$

and

$$f_\gamma = \frac{(\mathbb{P}[V_\lambda(s-1) < s\gamma] - \mathbb{P}[V_\lambda(s-1) > s\gamma])}{(\mathbb{P}[V_\lambda(s-1) = s\gamma - 1] + \mathbb{P}[V_\lambda(s-1) = s\gamma])}$$

Define $t = s\gamma$ and $\bar{\lambda} = 1 - \lambda$. Then,

$$\hat{y} \triangleq s\tilde{y} = \sum_{i=t}^{s-1} \binom{s-1}{i} \lambda^i \bar{\lambda}^{s-1-i} + f_\gamma \binom{s-1}{t-1} \lambda^{t-1} \bar{\lambda}^{s-t}$$

and

$$f_\gamma = \frac{g(\lambda)}{h(\lambda)}$$

where

$$g(\lambda) = \sum_{i=0}^{t-1} \binom{s-1}{i} \lambda^i \bar{\lambda}^{s-1-i} - \sum_{i=t+1}^{s-1} \binom{s-1}{i} \lambda^i \bar{\lambda}^{s-1-i}$$

and

$$h(\lambda) = \binom{s-1}{t-1}\lambda^{t-1}\bar{\lambda}^{s-t} + \binom{s-1}{t}\lambda^{t}\bar{\lambda}^{s-1-t}$$

Two useful identities are

$$\frac{d}{d\lambda}\sum_{i=0}^{t}\binom{s-1}{i}\lambda^{i}\bar{\lambda}^{s-1-i} = -(s-1)\binom{s-2}{t}\lambda^{t}\bar{\lambda}^{s-2-t}$$

$$\frac{d}{d\lambda}\sum_{i=t}^{s-1}\binom{s-1}{i}\lambda^{i}\bar{\lambda}^{s-1-i} = (s-1)\binom{s-2}{t-1}\lambda^{t-1}\bar{\lambda}^{s-1-t}$$

Take the derivative and set the result to zero. In the derivation, constants and factors which are λ or $\bar{\lambda}$ do not matter, so we can factor these out.

$$
\begin{aligned}
0 =& \frac{d}{d\lambda}\hat{y} \\
=& \frac{d}{d\lambda}\left(\sum_{i=t}^{s-1}\binom{s-1}{i}\lambda^{t}\bar{\lambda}^{s-1-i} + \frac{g(\lambda)}{h(\lambda)}\binom{s-1}{t-1}\lambda^{t-1}\bar{\lambda}^{s-t}\right) \\
=& (s-1)\binom{s-2}{t-1}\lambda^{t-1}\bar{\lambda}^{s-2-(t-1)} \\
&+ \frac{g(\lambda)}{h(\lambda)}\left((t-1)\binom{s-1}{t-1}\lambda^{t-2}\bar{\lambda}^{s-t} - (s-t)\binom{s-1}{t-1}\lambda^{t-1}\bar{\lambda}^{s-1-t}\right) \\
&+ \frac{g'(\lambda)h(\lambda) - h'(\lambda)g(\lambda)}{(h(\lambda))^2}\left(\binom{s-1}{t-1}\lambda^{t-1}\bar{\lambda}^{s-t}\right) \\
0 =& (h(\lambda))^2(s-1)\binom{s-2}{t-1}\lambda^{t-1}\bar{\lambda}^{s-2-(t-1)} \\
&+ g(\lambda)h(\lambda)\left((t-1)\binom{s-1}{t-1}\lambda^{t-2}\bar{\lambda}^{s-t} - (s-t)\binom{s-1}{t-1}\lambda^{t-1}\bar{\lambda}^{s-1-t}\right) \\
&+ (g'(\lambda)h(\lambda) - h'(\lambda)g(\lambda))\left(\binom{s-1}{t-1}\lambda^{t-1}\bar{\lambda}^{s-t}\right) \\
=& h(\lambda)\frac{s-1}{t}\binom{s-2}{t-1}\lambda^{t}\bar{\lambda}^{s-1-t} - h(\lambda)\frac{s-1}{s-t}\binom{s-2}{t}\lambda^{t}\bar{\lambda}^{s-1-t} \\
&+ g(\lambda)\frac{s-1}{t}\binom{s-2}{t-2}\lambda^{t-1}\bar{\lambda}^{s-t} - g(\lambda)\frac{s-1}{s-t}\binom{s-2}{t-1}\lambda^{t-1}\bar{\lambda}^{s-t} \\
&+ g(\lambda)\frac{s-1}{s-t}\binom{s-2}{t}\lambda^{t}\bar{\lambda}^{s-1-t} - g(\lambda)\frac{s-1}{t}\binom{s-2}{t-1}\lambda^{t}\bar{\lambda}^{s-1-t} \\
=& h(\lambda)\left(\binom{s-1}{t} - \frac{s-1-t}{s-t}\binom{s-1}{t}\right)\lambda^{t}\bar{\lambda}^{s-1-t} \\
&+ g(\lambda)\left(\frac{t-1}{t}\binom{s-1}{t-1} - \binom{s-1}{t-1}\right)\lambda^{t-1}\bar{\lambda}^{s-t} \\
&+ g(\lambda)\left(\frac{s-1-t}{s-t}\binom{s-1}{t} - \binom{s-1}{t-1}\right)\lambda^{t}\bar{\lambda}^{s-1-t} \\
=& h(\lambda)\frac{1}{s-t}\binom{s-1}{t}\lambda^{t}\bar{\lambda}^{s-t}\frac{1}{\bar{\lambda}} + g(\lambda)\frac{-1}{t}\binom{s-1}{t-1}\lambda^{t}\bar{\lambda}^{s-t}\frac{1}{\lambda} + g(\lambda)\frac{-1}{s-t}\binom{s-1}{t}\lambda^{t}\bar{\lambda}^{s-t}\frac{1}{\bar{\lambda}} \\
=& \left(\frac{h(\lambda)}{\bar{\lambda}} - \frac{g(\lambda)}{\lambda} - \frac{g(\lambda)}{\bar{\lambda}}\right)\frac{1}{s}\binom{s}{t}\lambda^{t}\bar{\lambda}^{s-t} \\
=& \lambda h(\lambda) - g(\lambda)
\end{aligned}
$$

Thus solutions of $\lambda h(\lambda) - g(\lambda) = 0$ must also be a solution to $\frac{d}{d\lambda}\hat{y} = 0$. Since $f_\gamma = \frac{g(\lambda)}{h(\lambda)} = \lambda$ also fits the constraint that $0 \leq f_\gamma \leq 1$, it must be that $f_\gamma = \lambda$. $\square$

Bibliography

[1] J. von Neumann, "Probabilistic logics and the synthesis of reliable organs from unreliable componenets", in C.E. Shannon and J. McCarthy (Eds.), *Automata Studies*, Princeton University Press, 1956, pp. 43-98.

[2] N. Pippenger, "On Networks of noisy gates," in *Proc. 26th Annu. Symp. Foundations Computer Science*, 1985, pp. 30-38.

[3] E.F. Moore, C.E. Shannon. "Reliable Circuits Using Less Reliable Relays".

[4] J. Heath, G.S.P. Kuekes, and R. Williams. "A defect tolerant computer architecture: Opportunities for nanotechnology". *Science*, 80:1716-1721, 1998.

[5] P. Bogdan, T. Dumitras, R. Marculescu. "Stochastic Communication: A New Paradigm for Fault Tolerant Networks-on-Chip".

[6] T. Leighton, C.E. Leiserson. "Wafer-Scale Integration of Systolic Arrays".

[7] K. Nikolic, A. Sadek, and M. Forshaw. "Fault-Tolerant techniques for nanocomputers".

[8] G. Norman, D. Parker, and M. Kwiatkowska. "Evaluation the reliability of defect-tolerant architectures for Nanatechnology with Probabilistic Model Checking".