Hash Families and Applications to t-Restrictions

by

Ryan Dougherty

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved August 2019 by the
Graduate Supervisory Committee:

Charles Colbourn, Chair
Andrzej Czygrinow
Stephanie Forrest
Andrea Richa

ARIZONA STATE UNIVERSITY

December 2019

ABSTRACT

The construction of many families of combinatorial objects remains a challenging problem. A t-restriction is an array where a predicate is satisfied for every t columns; an example is a perfect hash family (PHF). The composition of a PHF and any t-restriction satisfying predicate P yields another t-restriction also satisfying P with more columns than the original t-restriction had. This thesis concerns three approaches in determining the smallest size of PHFs.

Firstly, hash families in which the associated property is satisfied at least some number λ times are considered, called *higher-index*, which guarantees redundancy when constructing t-restrictions. Some direct and optimal constructions of hash families of higher index are given. A new recursive construction is established that generalizes previous results and generates higher-index PHFs with more columns. Probabilistic methods are employed to obtain an upper bound on the optimal size of higher-index PHFs when the number of columns is large. A new deterministic algorithm is developed that generates such PHFs meeting this bound, and computational results are reported.

Secondly, a restriction on the structure of PHFs is introduced, called *fractal*, a method from Blackburn. His method is extended in several ways; from homogeneous hash families (every row has the same number of symbols) to heterogeneous ones; and to distributing hash families, a relaxation of the predicate for PHFs. Recursive constructions with fractal hash families as ingredients are given, and improve upon on the best-known sizes of many PHFs.

Thirdly, a method of Colbourn and Lanus is extended in which they horizontally copied a given hash family and greedily applied transformations to each copy. Transformations of existential t-restrictions are introduced, which allow for the method to be applicable to any t-restriction having structure like those of hash families. A

genetic algorithm is employed for finding the “best” such transformations. Computational results of the GA are reported using PHFs, as the number of transformations permitted is large compared to the number of symbols. Finally, an analysis is given of what trade-offs exist between computation time and the number of t-sets left not satisfying the predicate.

To Nancy and Jeannine, with love. I miss you both.

ACKNOWLEDGMENTS

First, I would like to thank my doctoral advisor Dr. Charles Colbourn. His dedication to the subject and to push me beyond what I thought I was capable of at every step of the way was inspiring. He was able to help me whenever I was stuck on something, and to divert me back on track when I went astray. I am very happy he was my advisor, because I couldn't ask for a better one.

Second, I would like to thank my committee members: Dr. Andrzej Czygrinow, Dr. Stephanie Forrest, and Dr. Andrea Richa. I have taken a course from all three of these members, and all of them either directly or indirectly influenced the work in this dissertation. They were always wanting to accommodate to their busy schedules for scheduling defenses and meetings, and further inspire me to become as prolific an academic as they are.

Third, I would like to thank my friends and family (in alphabetical order): Bonnie, Cole, Ed, Erin, Ethan, Jackie, Jeannine, John, Julie, Mark, Nancy, and Quinn. Their love and support kept me going during the highest of highs, and the lowest of lows. I also want to thank the students in my classes for their encouragement and support; I especially thank Rachel for letting me serve on her undergraduate honors thesis committee.

Fourth, I would like to thank CIDSE for partially supporting the work in this dissertation, and allowing me to be the instructor of record for 8 courses throughout my Ph.D. (especially since one of my publications resulted from TA work). I would like to also thank ACM and the Cumberland Organizers for partially funding travel to the GECCO 2019 and 30th+31st Cumberland Conferences to present some of this research.

TABLE OF CONTENTS

LIST OF TABLES

Table	Page

LIST OF FIGURES

Figure Page

Figure Page

GLOSSARY

Abstract Simplicial Complex : an *abstract simplicial complex*, $\mathcal{A}$, is a family of non-empty finite subsets of a set Γ that is closed under non-empty subsets (page 8).

Covering Array : when for every t-set $\{c_1, \ldots, c_t\}$ of columns, the demand $(\Delta_{c_1} \times \cdots \times \Delta_{c_t}, \forall)$ is to be met, the array is a *mixed-level covering array.* A covering array is when all of the Δ_{c_i} have the same cardinality (page 13).

Covering Perfect Hash Family : an $N \times k$ array where each entry is from $\mathsf{V}_{t,q}$ (set of representatives of permutation vectors), and for every t distinct columns $c_1, \cdots, c_t$, there is a row ρ for which the $t \times t$ matrix of the t corresponding entries is nonsingular over $\mathbb{F}_q$ (page 11).

Distributing : choose an integer t, and form the set $\mathcal{M}_t$ of multisets whose elements contain nonnegative integers, for which the sums of each element in a multiset sum to t. When $\mathcal{W}$ consists of all partitions in $\mathcal{M}_t$ containing s parts, a $\mathcal{W}$-separating hash family is (t, s)-*distributing* (page 10).

Existential Restriction : an existential t-restriction is one that has all of the T_i being $\exists$ (page 98).

Fractal : a hash family is fractal if the removal of any row yields another hash family with strength at least 1 less than the original strength (or the number of parts is reduced by 1) (page 76).

Genetic Algorithm : an algorithm that maintains a population P and tries to find individuals in P that maximize a given fitness function f via operators (namely mutation and crossover) that are repeatedly applied to the individuals. At each iteration, some of the members in P are removed before the next "generation" of individuals occurs (page 95).

Heterogeneous : an array is heterogeneous if the number of symbols for some rows i, j $(i \neq j)$ is different (i.e., not homogeneous) (page 9).

Homogeneous : an array is homogeneous if the number of allowed symbols for each row is the same (page 9).

Index : for any array λ-satisfying a t-restriction, λ is the index of the array (page 9).

Linear : a transversal design is linear if it is constructed by taking each polynomial $a_0 + a_1 y + \cdots + a_{s-1} y^{s-1}$ of degree $s - 1$ with coefficients from $\mathbb{F}_q$, form a block that contains element (b, z) whenever $z \in X$ and (1) $b = a_0$ when $z = \infty$, or (2) $b = a_0 + a_1 z + \cdots + a_{s-1} z^{s-1}$ otherwise (all arithmetic performed in $\mathbb{F}_q$) (page 25).

Mixed : an array is mixed if the number of allowed symbols for some columns i, j $(i \neq j)$ is different (i.e., not uniform) (page 9).

Monotone Restriction : a monotone t-restriction has all of the T_i being equal (so either $T_1 = \cdots = T_\chi = \exists$ or $T_1 = \cdots = T_\chi = \forall$) (page 9).

Partition Covering Array : an $N \times s$ array such that for every nonempty ordered partition of size i of $[t]$ for every valid i, every choice of i columns fully covers that partition at least λ times (page 29).

Partitioned Ordered Design : written $\mathsf{POD}_\lambda(N; s, (w_1, \cdots, w_m), (r_1, \cdots, r_m))$, it is an $N \times s$ array in which (1) $w_i < w_j$ for all $i < j$, and (2) every p-tuple formed from $(w_1, \cdots, w_m)$ by repeating each w_i exactly r_i times, in any order, is fully covered at least λ times in the design (page 29).

Perfect Hash Family : an (s, s)-distributing hash family. In other words, it is an $N \times k$ array such that for every t columns, they are separated by λ rows (i.e., λ rows have all distinct symbols in those t columns) (page 10).

Perfect Hash Family Column Number : the largest number of columns for which a perfect hash family exists (page 15).

Perfect Hash Family Number : the smallest number of rows for which a perfect hash family exists (page 15).

Perfect Heterogeneous Hash Family : a perfect hash family where each row i is over an alphabet of v_i symbols (page 10).

Resolvable Balanced Incomplete Block Design : a set of v points X, b subsets of X each with k points, every point occurs in r blocks, and every pair of points occurs in λ blocks (page 27).

Restriction : a *t-restriction* is a χ-tuple $\mathcal{T} = ((\mathcal{P}_1, T_1), \cdots, (\mathcal{P}_\chi, T_\chi))$, where $\mathcal{P}_i \subseteq \Delta^t$ and $T_i \in \{\exists, \forall\}$. Each set $\mathcal{P}_i$ is a *demand*. For each $\mathcal{P}_i$, if $T_i = \exists$, then at least λ rows of A contains some element of $\mathcal{P}_i$; if $T_i = \forall$, then for each element of $\mathcal{P}_i$, at least λ rows contain that element. Let $\partial^i(\mathcal{S})$ be the set of $\binom{t}{i}$ sets of $(t-i)$-tuples, obtained by deleting the i chosen columns from each $s \in \mathcal{S}$ (page 9).

Satisfies : a set of t columns in a t-restriction λ-satisfies a demand $(\mathcal{P}_i, T_i)$ if (1) there exist λ rows for which some element(s) of $\mathcal{P}_i$ appears when $T_i = \exists$, or (2) all elements in $\mathcal{P}_i$ appear in these columns at least λ times when $T_i = \forall$ (page 9).

Separating Hash Family : a $\mathcal{W}$-separating hash family meets the following condition: when $C = \{c_1, \cdots, c_t\} \subseteq \binom{[k]}{t}$ and $W_1, \cdots, W_s$ is a partition of C with $\{|W_1|, \cdots, |W_s|\} \in \mathcal{W}$, define $\mathcal{D} = \{(y_1, \ldots, y_t) \in \Delta_{c_1} \times \cdots \times \Delta_{c_t} : y_c = y_{c'}$ only if c, c' belong to the same class of $W\}$. Then the demand $(\mathcal{D}, \exists)$ is met (page 10).

Strength : aor any array satisfying a t-restriction, t is the strength of the array (page 9).

Transformation : a function ϕ such that A satisfies a given t-restriction $\mathcal{T}$ if and only if $\phi(A)$ also does, and both $A, \phi(A)$ have the same number of rows and columns (page 98).

Uniform : an array is uniform if the number of allowed symbols for each column is the same (page 9).

Universal Restriction : a universal t-restriction is one that has all of the T_i being $\forall$ (page 98).

NOTATIONS

ASC : an abstract simplicial complex (page 8).

$\mathsf{CA}_\lambda(N;t,k,v)$: a covering array with N rows, k columns, v symbols, and strength t (page 13).

$\mathsf{CPHF}_\lambda(N;k,q,t)$: a covering perfect hash family with N rows, k columns, q symbols (a prime power), and strength t (page 12).

$\mathsf{DHF}_\lambda(N;k,v,t,s)$: a distributing (homogeneous) hash family with N rows, k columns, v symbols, strength t, and (at most) s parts (page 11).

$\mathsf{DHHF}_\lambda(N;k,(v_1,\cdots,v_N),\mathcal{W})$: a distributing (heterogeneous) hash family with N rows, k columns, v_i symbols for each row i, strength t, and (at most) s parts (page 11).

$\mathsf{MCA}_\lambda(N;t,k,(v_1,\cdots,v_k))$: a (mixed-level) covering array with N rows, k columns, v_i symbols for each column i, and strength t (page 13).

$\mathsf{PaHF}_\lambda(N;k,v,t,s)$: a partitioning (homogeneous) hash family with N rows, k columns, v symbols, strength t, and (at most) s parts (page 14).

$\mathsf{PCA}_\lambda(N;t,s)$: a partition covering array with N rows, s columns (page 29).

$\mathsf{PHF}_\lambda(N;k,v,t)$: a perfect (homogeneous) hash family with N rows, k columns, v symbols, and strength t (page 10).

$\mathsf{PHFK}_\lambda(N,v,t)$: the perfect (homogeneous) hash family (column) number (page 15).

$\mathsf{PHFN}_\lambda(k,v,t)$: the perfect (homogeneous) hash family (row) number (page 15).

$\mathsf{PHHF}_\lambda(N;k,(v_1,\cdots,v_N),t)$: a perfect (heterogeneous) hash family with N rows, k columns, v_i symbols for each row i, and strength t (page 10).

$\mathsf{PHHFK}_\lambda(N,(v_1,\cdots,v_N),t)$: the perfect (heterogeneous) hash family (column) number (page 15).

$\mathsf{POD}_\lambda(N;s,(w_1,\cdots,w_m),(r_1,\cdots,r_m))$: a partitioning ordered design of type $(w_1,\cdots,w_m)$ and replication $(r_1,\cdots,r_m)$ with N rows and s columns (page 30).

$\mathsf{PODN}_\lambda(s,(w_1,\cdots,w_m),(r_1,\cdots,r_m))$: the partitioning ordered design (row) number (page 31).

$\mathsf{SCPHF}_\lambda(N;k,q,t)$: a Sherwood covering perfect hash family with N rows, k columns, q symbols (a prime power), and strength t (page 12).

$\mathsf{SHHF}_\lambda(N;k,(v_1,\cdots,v_N),\mathcal{W})$: a $\mathcal{W}$-separating (heterogeneous) hash family with N rows, k columns, v_i symbols for each row i, and for each $W \in \mathcal{W}$, every partition of $\sum_{w_i \in W} w_i$ columns is separated in at least λ rows (page 10).

$\mathsf{TRA}_\lambda(N,k,\mathcal{H},(v_1,\cdots,v_N),(x_1,\cdots,x_k),\mathcal{T})$: a t-restriction array with N rows, k column, t-uniform hypergraph $\mathcal{H}$, v_i symbols for each column i, x_i symbols for each row i, and restriction $\mathcal{T}$ (page 9).

Chapter 1

INTRODUCTION

This thesis is about a set of combinatorial objects called *hash families*, and their applications to a larger umbrella of objects, called *t-restrictions*. In this chapter, we motivate the study of such families by discussing applications of t-restrictions. The goal of the thesis is to achieve a better understanding of the structure of hash families. To obtain a better appreciation of the mathematics behind them, we begin with an informal discussion, and then discuss all necessary formal background in Chapter 2. At the end of this chapter, we discuss three research problems undertaken, as well as the organization of the rest of the thesis.

1.1 Representative Problems

1.1.1 Interaction Testing

Imagine we have a large-scale software system that needs to be verified for correctness; usually a software developer determines this by designing a test suite that has desirable properties; examples include code coverage (every line executed by at least one test case), and testing every possible set of inputs. The latter is often impossible due to having infinitely many possible inputs, such as a prime number verifier: given a positive integer n, determine if n is prime or not. There are infinitely many such integers, so exhaustive testing is completely intractable. Therefore, most test suites do not strive for all inputs to be tested, but rather a representative sample; this choice of testing can potentially lead to faults within the system.

Suppose further in our system each component is individually tested ("unit testing"), in that in isolation, each of the components functions properly. Problems may then arise when multiple components operate simultaneously; an example is a smartphone that is downloading an application and loading a webpage at the same time. Here, the network is being used by two different components. We then want to model any *interactions* that may arise between different components. Empirically, most faults involve only a small number of components [54].

Covering arrays were developed to help model this type of interaction testing problem, in that all possible interactions of components of size at most a specified *strength* t appear in the array at least once. Each of the components C_i constitutes a number of possible inputs v_i (these inputs are the *levels* of component C_i); we assume here that each component's input space is discretized and has finitely many values. Then if we want to model an interaction of size t among components $C_{i_1}, \cdots, C_{i_t}$, then all of the $v_{i_1} \times v_{i_2} \times \cdots \times v_{i_t}$ possible valuations of these t components needs to be tested. The array itself is an $N \times k$ array (i.e., N rows and k columns), where each of the columns corresponds to a component, and each of the rows corresponds to a test of the system.

Here, covering arrays are to *detect the existence of a fault* (if there is one). Suppose when running components $C_{i_1}, \cdots, C_{i_t}$ with values $v_1, \cdots, v_t$, a fault arises within the system. Because the array covers all interactions, this one appears in some row of the array, say row ρ. Then if one were to execute each of the tests in the covering array on the system, then executing test ρ will notify the tester that a fault exists. Note that covering arrays only detect if a fault exists, rather than determine which components and values cause the fault to occur; the latter is the subject of detecting and locating arrays [37].

We give a concrete example of a covering array in Figure 1.1. It is a covering array with 9 rows, 4 components, each having 3 levels, and strength 2. Note that 9 rows are required, so this covering array has the smallest number of rows possible. From

Browser	OS	Connection	Printer
Safari	Windows	LAN	Local
Safari	Linux	ISDN	Networked
Safari	macOS	PPP	Screen
IE	Windows	ISDN	Screen
IE	macOS	LAN	Networked
IE	Linux	PPP	Local
Chrome	Windows	PPP	Networked
Chrome	Linux	LAN	Screen
Chrome	macOS	ISDN	Local

Figure 1.1: A Covering Array with 9 Rows, 4 Components, Each Having 3 Levels, and Strength 2.

a testing point-of-view, it is useful to execute fewer tests while still maintaining the coverage guarantee. Much research regarding covering arrays has been to minimize the number of rows, while fixing the number of columns, levels, and strength [23, 66, 67]. Covering arrays also have applications in testing advanced materials [22], regulating gene expression [70], learning boolean functions [40], and more recently in malware analysis [55].

1.2 Hash Families

Suppose that there are k items, and each is assigned one of v values. Our objective is to ensure that each set of t items receives t different values; when this occurs, the t items are *separated*. Evidently if $v \geq k$, each item can be assigned a value that is different from all others assigned, so that every set of t items is separated. However, when $v < k$, some two items necessarily receive the same value; then any t-set containing these two cannot be separated. When this occurs, suppose that

N assignments of values to items are chosen, rather than one. Then one can ask: how small can N be so that every t-set of items is separated in at least 1 of the assignments? This easily stated combinatorial question is challenging, and many open problems remain despite substantial research effort. It is an important question as well, with applications described next.

Mehlhorn [58] originally examined this question to provide an efficient way to store and retrieve frequently used information; in that context, the assignment of values to the items is treated as a *hash function* [39], and hence the question is phrased as one about families of hash functions. Applications to derandomization [6], circuit complexity [61] , and cryptography [17, 49, 74] arose. Subsequently, Stinson, Trung, and Wei [75] established applications of such families (with $\lambda = 1$) to construct numerous other combinatorial objects, such as separating systems, key distribution patterns, cover-free families, and secure frameproof codes. A general strategy, column replacement, has extended their range of applications into testing and measurement [27] and compressive sensing [31], among others.

1.3 t-Restrictions

We note that the previous two problems are variations upon a common theme; there is an array of symbols, such that for any t columns, there is an associated set of t-tuples X, and either (1) some row of these t columns contains some element from X, in the case of hash families; or (2) all elements of X appear in some row, in the case of covering arrays. Any array that falls under this definition we call a *t-restriction*; note that instead of having one predicate being satisfied, we can have several, and each can be either (1) or (2) above. We call these predicates *demands* of the restriction. For example, suppose 3 columns of a covering array all have levels $\{0, 1, 2\}$; of course, all 27 tuples need to appear in the array. Suppose further that only one 3-tuple that has a 0 as its first element needs to be tested, instead of all of them. We now have two demands: X_1, which consists of all 3-tuples where 0 is the first element, and X_2, which consists of all other 3-tuples. Also, the first demand only requires that some row contain some element of X_1, whereas all elements of X_2 need to appear. Here, $|X_1| = 3^2 = 9$, so only 19 instead of 27 tuples need to appear in each 3-set of columns. The relevance of such an object is that a software tester can eliminate, based on knowledge of the system, inputs that are guaranteed never to appear during normal execution of the system.

One notable aspect about perfect hash families is that a simple construction, using an array satisfying a t-restriction with k columns, and a perfect hash family with $m > k$ columns, yields an array with m columns satisfying the same t-restriction (see Theorem 2.1). Furthermore, the corresponding number of times the predicate is satisfied of the resulting t-restriction is the product of those for the original t-restriction and the perfect hash family. For example, if we have a covering array in which each interaction appears at least 3 times, and a perfect hash family (with matching parameters) that separates every t-set of columns at least 4 times, then the resulting covering array will cover each interaction at least 12 times. Interaction testing benefits from having more coverage, most notably in domains where the environment surrounding the system is not fixed, or the system is not completely deterministic.

1.4 Summary of Contributions

The contribution of the thesis is a greater understanding of the structure and generation of certain t-restrictions, when we generalize them beyond commonly used parameters. As far as we are aware, no publications exist in the investigation of hash families with the separation condition requiring every t-set of items being separated in strictly more than 1 of the assignments.

We investigate upper bounds on the sizes of hash families with a given separation requirement $\lambda > 1$, through direct constructions, a new recursive construction exploiting higher separation requirements, probabilistic and asymptotic analyses on upper bounds, and a new polynomial-time constructive algorithm to find such hash families meeting these upper bounds. The recursive construction improves on the sizes of perfect hash families when the strength t is "small" for some parameters. The analyses improve significantly on the "simple" method of vertically adjoining copies of the original array. And finally, the constructive algorithm uses estimates on the number of rows needed, which provides a guarantee on an upper bound for the number of rows that will be produced (and often, this estimate is improved significantly).

Motivated by the composition construction mentioned earlier, we develop the notion of *fractal* hash families, which yield a new recursive construction that produces hash families with "few" rows that improve on the best-known sizes of existing hash families. This method extends one of Blackburn [15], wherein only asymptotics are investigated, in that our methods explicitly produce the hash families and generalize the hash family considered. More than 2,500 individual parameter sets in the perfect hash family tables [45].

Finally, we develop a genetic algorithm that attempts to construct t-restrictions that generalizes a method of Colbourn and Lanus [33]. The algorithm horizontally adjoins copies of an existing array, such that fewer t-sets of columns need to be checked. Furthermore, the representation of each individual in the genetic algorithm's population is much smaller than if we are to apply existing methods to hash families. We show that the method always finds individual arrays faster than existing methods, and these arrays have higher "fitness" than the ones produced by other methods (i.e., more t-sets of columns are separated).

1.5 Organization of the Thesis

In Chapter 2, we introduce formal notation for all objects discussed previously that will be used in subsequent chapters. In Chapter 3, we develop construction techniques for hash families with arbitrary redundancy, in which the separation requirement is achieved at least a certain number of times. In Chapter 4, we consider hash families with “few” rows, as well as guaranteeing a structure within the array that yields hash families with rows smaller than the associated strength to be constructed, which often have the best-known number of columns. In Chapter 5, we develop a genetic algorithm to generate t-restrictions with many columns, given a “starter” ingredient with few columns, by exploiting the structure of a given t-restriction; this algorithm is general in that any restriction with a form similar to that of a hash family, and any initial array meeting the restriction can be selected. And finally, in Chapter 6 we give our conclusions, and provide many future research directions and open problems.

Chapter 2

BACKGROUND

In this chapter we establish the notation needed regarding hash families and t-restrictions. Theorem 2.1 illustrates an important connection between perfect hash families, specifically, and arbitrary t-restrictions, in that the latter can be constructed from the former and a "smaller" restriction. This motivates the study of perfect hash families and a combinatorial analysis of their structure. Then, we briefly mention some existing work on the sizes and structure of perfect hash families that is relevant later in the thesis. Any background material not mentioned in this chapter that is also relevant in other chapters will be in them instead.

2.1 Terminology

In order to develop and extend these ideas formally, we extend the presentation in [27], employing the very general language of t-restrictions [5]. Let N, k, $v_1, \cdots, v_N$, $x_1, \cdots, x_k$, t, and λ be positive integers. An *abstract simplicial complex*, $\mathcal{A}$, is a family of non-empty finite subsets of a set Γ that is closed under non-empty subsets; the *dimension* of an ASC, $\mathsf{dim}(\mathcal{A})$, is the maximum of $|X| - 1$, for all $X \in \mathcal{A}$. Let $\mathcal{H}$ be an abstract simplical complex on k vertices such that the maximum cardinality of any set in $\mathcal{H}$ is t; label the vertices of $\mathcal{H}$ as $c_1, \cdots, c_k$. Let Σ_i be a v_i-ary alphabet not containing $\star$ for all $1 \leq i \leq N$, and let Δ_j be an x_j-ary alphabet also not containing $\star$ for all $1 \leq j \leq k$. Define an $N \times k$ array A in which the ith row of A contains symbols from $\Sigma_i \cup \{\star\}$, and the jth column of A contains symbols from $\Delta_j \cup \{\star\}$. If there exist i, j for which $\Sigma_i \cap \Delta_j = \emptyset$, the entry in this cell must be $\star$. Let $\Delta = \bigcup_{j=1}^{k} \Delta_j$. A *$t$-restriction* is a χ-tuple $\mathcal{T} = ((\mathcal{P}_1, T_1), \cdots, (\mathcal{P}_\chi, T_\chi))$, where $\mathcal{P}_i \subseteq \Delta^t$ and $T_i \in \{\exists, \forall\}$.

Each set $\mathcal{P}_i$ is a *demand*. Here, we denote t to be the *strength* of the array. For each $\mathcal{P}_i$, if $T_i = \exists$, then at least λ rows of A contains some element of $\mathcal{P}_i$; if $T_i = \forall$, then for each element of $\mathcal{P}_i$, at least λ rows contain that element. Let $\partial^i(\mathcal{S})$ be the set of $\binom{t}{i}$ sets of $(t-i)$-tuples, obtained by deleting the i chosen columns from each $s \in \mathcal{S}$. An array $\mathsf{A} = (a_{ij})$ λ-*satisfies* a given $\mathcal{P}_i$ and T_i if and only if for all $0 \le j \le t$ and any set $S \in \mathcal{H}$,

1. if $T_i = \exists$, then for each $\mathcal{P} \in \partial^j(\mathcal{P}_i)$, there exist λ rows $1 \le r_1 < \cdots < r_\lambda \le N$ such that $(a_{r_\ell, c_{i_1}}, \cdots, a_{r_\ell, c_{i_{|S|}}}) \in \mathcal{P}$ for all $1 \le \ell \le \lambda$ and $S \in \mathcal{H}$ when $|S| = t - j$; or

2. if $T_i = \forall$, then for each $\mathcal{P} \in \partial^j(\mathcal{P}_i)$, and for all $(\sigma_1, \cdots, \sigma_{t-j}) \in \mathcal{P} \cap \prod_{m=1}^{t-j} \Delta_{c_m}$, there exist λ rows $1 \le r_1 < \cdots < r_\lambda \le N$ such that $(a_{r_\ell, c_{i_1}}, \cdots, a_{r_\ell, c_{i_{|S|}}}) = (\sigma_1, \cdots, \sigma_{|S|})$ for all $1 \le \ell \le \lambda$ and $S \in \mathcal{H}$ when $|S| = t - j$.

If the array λ-satisfies each of the given $\mathcal{P}_i$ and T_i, then the array λ-*satisfies* $\mathcal{T}$. If an array A on N rows and k columns (and corresponding symbol set cardinalities for rows and columns) λ-satisfies a t-restriction $\mathcal{T}$, denote it by $\mathsf{TRA}_\lambda(N, k, \mathcal{H}, (v_1, \cdots, v_N), (x_1, \cdots, x_k), \mathcal{T})$. For any such TRA_λ, we denote λ to be the *index* of the array. If $v_1 = \cdots = v_N$, A is *uniform*; otherwise, it is *mixed*. If $x_1 = \cdots = x_k$, A is *homogeneous*; otherwise, it is *heterogeneous*. When $\lambda = 1$, we omit it from the notation. When $T_1 = \cdots = T_\chi$, $\mathcal{T}$ is a monotone t-restriction. Most literature has concentrated on monotone t-restrictions with $\mathcal{H}$ being the hypergraph containing all possible hyperedges of size at most t on k vertices, and $\lambda = 1$.

This framework is very general, and it encompasses a number of well-studied combinatorial arrays. We establish more restrictive notation for some of them next. Choose an integer t, and form the set $\mathcal{M}_t$ of multisets whose elements contain non-negative integers, for which the sums of each element in a multiset sum to t. Let

$\mathcal{W} \subseteq \mathcal{M}_t$. A $\mathcal{W}$*-separating hash family* meets the following condition: when $C = \{c_1, \cdots, c_t\} \subseteq \binom{[k]}{t}$ and $W_1, \cdots, W_s$ is a partition of C with $\{|W_1|, \cdots, |W_s|\} \in \mathcal{W}$, define $\mathcal{D} = \{(y_1, \ldots, y_t) \in \Delta_{c_1} \times \cdots \times \Delta_{c_t} : y_c = y_{c'}$ only if c, c' belong to the same class of $W\}$. Then the demand $(\mathcal{D}, \exists)$ is met. When each demand is the stricter requirement that $\mathcal{D} = \{(y_1, \ldots, y_t) \in \Delta_{c_1} \times \cdots \times \Delta_{c_t} : y_c = y_{c'}$ if and only if c, c' belong to the same class of $W\}$, the hash family is $\mathcal{W}$*-partitioning*. When $\mathcal{W}$ consists of all partitions in $\mathcal{M}_t$ containing s parts, a $\mathcal{W}$-separating hash family is (t, s)*-distributing*. In both cases, when $\mathcal{W}$ contains a single set $W = \{w_1, \ldots, w_s\}$, the family is separating (or partitioning) of *type* $\{w_1, \ldots, w_s\}$.

Of primary concern here are the (t, s)-distributing hash families with $s = t$. Such a family is a *perfect hash family*. In order to refer to objects of this type, we employ standard notation. A perfect heterogeneous hash family is denoted as a $\mathsf{PHHF}_\lambda(N; k, (v_1, \cdots, v_N), t)$, and a homogeneous one is written as a $\mathsf{PHF}_\lambda(N; k, v, t)$. If a PHHF_λ A λ-satisfies a given $\mathcal{P}_i$, then it λ*-separates* these columns.

An example of a (homogeneous) $\mathsf{PHF}_1(6; 12, 3, 3)$ is given in Figure 2.1. It is a 6×12 array (6 rows, 12 columns) on the three symbols $\{0, 1, 2\}$, in which every 3-set of columns is 1-separated. For the 6×3 subarray involving columns $8, 9$, and 10, only the last row consists of distinct symbols. Also, 148 of the 3-sets are exactly 1-separated; 44 are exactly 2-separated; 19 are exactly 3-separated; 4 are exactly 4-separated; and none are 5 or 6-separated. There is no $\mathsf{PHF}(5; 12, 3, 3)$ [9], so this array has the fewest possible rows.

The notation $\mathsf{SHHF}(N; k, (v_1, \cdots, v_N), \{w_1, \cdots, w_s\})$ is used for a separating hash family. More simply $\mathsf{SHF}(N; k, v, \{w_1, \cdots, w_s\})$ is used when it is homogeneous. Figure 2.2 gives an example of a (homogeneous) $\mathsf{SHF}(3; 16, 4, \{1, 2\})$. It is a 3×16 array on the four symbols $\{1, 2, 3, 4\}$ that is not a perfect hash family, because columns 11, 15, and 16 are separated by none of the three rows. However, in the 3×3 subarray

$$\begin{array}{c|cccccccccccc}
 & & & & & & & & \downarrow & \downarrow & \downarrow & & \\
\hline
 & 0 & 1 & 2 & 2 & 1 & 2 & 2 & 0 & 1 & 1 & 0 & 0 \\
 & 0 & 2 & 1 & 0 & 2 & 2 & 2 & 1 & 0 & 1 & 2 & 1 \\
 & 1 & 0 & 0 & 2 & 2 & 2 & 1 & 1 & 2 & 1 & 0 & 2 \\
 & 2 & 0 & 1 & 1 & 2 & 0 & 2 & 0 & 1 & 1 & 2 & 1 \\
 & 2 & 0 & 2 & 1 & 2 & 1 & 0 & 2 & 2 & 1 & 1 & 0 \\
\rightarrow & 2 & 0 & 1 & 2 & 1 & 1 & 2 & \boxed{2} & \boxed{0} & \boxed{1} & 2 & 1
\end{array}$$

Figure 2.1: A $\mathsf{PHF}_1(6; 12, 3, 3)$.

$$\begin{array}{c|cccccccccccccccc}
 & & & & & & & & & & & \downarrow & & & & \downarrow & \downarrow \\
\hline
 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 \\
 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 \\
\rightarrow & 1 & 2 & 3 & 4 & 2 & 1 & 4 & 3 & 3 & 4 & \boxed{1} & 2 & 4 & 3 & \boxed{2} & \boxed{1}
\end{array}$$

Figure 2.2: A $\mathsf{SHF}(3; 16, 4, \{1, 2\})$.

consisting of these three columns, each of the three $\{1, 2\}$-separations is accomplished by a row.

A distributing hash family is denoted by $\mathsf{DHHF}(N; k, (v_1, \cdots, v_N), t, s)$; a homogeneous DHHF is a $\mathsf{DHF}(N; k, v, t, s)$. Figure 2.3 gives a (heterogeneous) $\mathsf{DHHF}(10; 13, \mathbf{v}, 5, 2)$ with $\mathbf{v} = (9, 9, 9, 3, 3, 3, 3, 4, 5, 2)$. Often one uses an exponential notation that indicates the repetition in the exponent: $\mathbf{v} = (9^3 3^4 4^1 5^1 2^1)$.

Let q be a prime or prime power, and $\mathbb{F}_q$ the finite field of order q. Let $\mathsf{R}_{t,q} = \{r_0, \cdots, r_{q^t-1}\}$ be the set of all row vectors of length t with entries in $\mathbb{F}_q$. Let $\mathsf{T}_{t,q}$ be the set of all column vectors of length t with entries in $\mathbb{F}_q$, excluding the 0-vector. We call $x \in \mathsf{T}_{t,q}$ a *permutation vector*. Consider a set of t vectors $X = \{x_1, \cdots, x_t\} \subseteq \mathsf{T}_{t,q}$. Form an array A by setting $A_{i,j}$ to be the product of r_i, x_j; A is a $q^t \times t$ matrix where every row is distinct if and only if the $t \times t$ matrix B formed by horizontally

$$
\begin{array}{c|ccccccccccccc}
 & & & & & \downarrow & \downarrow & & & & \downarrow & & \downarrow & \downarrow \\
\hline
 & 6 & 7 & 8 & 3 & 4 & 0 & 2 & 2 & 3 & 0 & 5 & 1 & 1 \\
 & 3 & 1 & 1 & 7 & 2 & 6 & 8 & 4 & 3 & 0 & 2 & 0 & 5 \\
 & 8 & 5 & 1 & 4 & 2 & 3 & 2 & 6 & 7 & 0 & 1 & 3 & 0 \\
 & 0 & 2 & 0 & 2 & 2 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 0 \\
 & 0 & 0 & 2 & 1 & 1 & 1 & 2 & 0 & 0 & 2 & 2 & 0 & 1 \\
\rightarrow & 1 & 1 & 2 & 2 & \boxed{2} & \boxed{0} & 1 & 0 & 0 & \boxed{2} & 1 & \boxed{0} & \boxed{0} \\
 & 1 & 0 & 1 & 2 & 0 & 0 & 2 & 0 & 0 & 1 & 2 & 2 & 1 \\
 & 1 & 1 & 0 & 1 & 0 & 3 & 2 & 0 & 2 & 0 & 1 & 0 & 2 \\
 & 0 & 0 & 3 & 0 & 1 & 0 & 0 & 2 & 4 & 0 & 0 & 1 & 0 \\
 & 0 & \star & \star & \star & \star & 1 & \star & \star & 1 & \star & \star & 0 & 1
\end{array}
$$

Figure 2.3: A $\mathsf{DHHF}(10; 13, \mathbf{v}, 5, 2)$ with $\mathbf{v} = (9^3 3^4 4^1 5^1 2^1)$.

juxtaposing $x_1, \cdots, x_t$ is non-singular over the field. Note that the matrix B formed here cannot contain the 0-vector.

Let $\langle x \rangle = \{\mu x : \mu \in \mathbb{F}_q \setminus 0\}$. If x is not the 0-vector, the *representative* of $\langle x \rangle$, r_x, is the unique vector where the first nonzero coordinate is the multiplicative identity of $\mathbb{F}_q$. Let $\mathsf{V}_{t,q}$ be the set of representatives of all $x \in \mathsf{T}_{t,q}$. Let $\mathsf{U}_{t,q}$ be the set of vectors in $\mathsf{V}_{t,q}$ with first coordinate being nonzero (namely, the multiplicative identity). A *covering perfect hash family* is a 4-tuple $\mathsf{CPHF}_\lambda(N; k, q, t)$ which is an $N \times k$ array where each entry is from $\mathsf{V}_{t,q}$, and for every t distinct columns $c_1, \cdots, c_t$, there is a row ρ for which the corresponding matrix formed above is nonsingular (we say that $c_1, \cdots, c_t$ are *covered* when this occurs). We say that the array is a *Sherwood covering perfect hash family*, written $\mathsf{SCPHF}_\lambda(N; k, q, t)$ if every entry in the array is from $\mathsf{U}_{t,q}$. In this setting, $\mathcal{P}_i$ is the set of nonsingular matrices over $\mathbb{F}_q$, and $T_i = \exists$. Sherwood et al. [71] show that if a $\mathsf{SCPHF}(N; k, v, t)$ exists, then a $\mathsf{CA}(N(v^t - v) + v; t, k, v)$ exists.

Colbourn et al. [34] derive asymptotics for CPHFs along with many computational results and variants that provide many of the best-known results for covering arrays for strengths $3 \le t \le 6$ and $3 \le v \le 25$.

Hash families in general, and perfect hash families in particular, play a central role in the construction of arrays that satisfy various t-restrictions. Indeed, they form the essential ingredients in a general technique known as *composition* or *column replacement*, which we describe next.

Theorem 2.1. *Suppose there exist:*

1. A, *a* $\mathsf{PHF}_\chi(M; \ell, k, t)$*; and*

2. B, *a* $\mathsf{TRA}_\lambda(N; k, \mathcal{H}, (v_1^{s_1} \cdots, v_\rho^{s_\rho}), x^k, \mathcal{T})$ *with* $N = \sum_{i=1}^{\rho} s_i v_i$ *and* $\mathcal{H} = \binom{[k]}{t}$.

Construct an $NM \times \ell$ *array,* C*, by replacing each symbol* γ *in* A *by the column indexed by* γ *in* B*. Then* C *is a*

$$\mathsf{TRA}_{\chi\lambda}(NM; \ell, \mathcal{H}', ((M \cdot v_1)^{s_1} \cdots, (M \cdot v_\rho)^{s_\rho}), x^\ell, \mathcal{T})$$

with $\mathcal{H}' = \binom{[\ell]}{t}$.

It is possible to extend this construction for when $\mathcal{H} \neq \binom{[k]}{t}$ and the PHF_χ does not separate every t-set, but we content ourselves with the generality developed here. Construction 2.1 provides strong motivation for the study of perfect hash families, as it underlies the easy generation of 'large' arrays meeting t-restrictions. We outline one example of this, introducing a well-studied t-restriction that employs universal quantification (i.e., a *universal t-restriction*). When for every t-set $\{c_1, \ldots, c_t\}$ of columns, the demand $(\Delta_{c_1} \times \cdots \times \Delta_{c_t}, \forall)$ is to be met, the array is a *mixed-level covering array*, denoted by $\mathsf{MCA}(N; t, (v_1, \cdots, v_k))$; when the array is homogeneous, it is a *covering array*, denoted by $\mathsf{CA}(N; t, k, v)$. In any $\mathsf{CA}(N; k, v, t)$, symbols can

0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	0	0	0	0	1
1	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	0	0	0
0	1	1	0	0	1	0	0	1	0
0	0	1	0	1	0	1	1	1	0
1	1	0	1	0	0	1	0	1	0
0	0	0	1	1	1	0	0	1	1
0	0	1	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	1	1	1
0	1	0	0	0	1	1	1	0	1

Figure 2.4: A $\mathsf{CA}(13; 3, 10, 2)$.

be permuted *in each column independently* so that the first row consists entirely of a single symbol. This yields a *constant row*, and when the CA has been modified in this way, it is *standardized*. Figure 2.4 gives an example of a standardized $\mathsf{CA}(13; 3, 10, 2)$.

In [26], a restriction on DHHFs is applied to gain an additional improvement on Theorem 2.1 when the TRA is a covering array. *Partitioning* hash families are distributing hash families except not only for any partition of t-columns into s parts (possibly empty) the entries in any two different parts are pairwise disjoint, but the symbols in each part are all equal; denote it by a $\mathsf{PaHF}(N; k, v, t, s)$. It is shown there that a $\mathsf{DHF}(N; k, 2, t, 2)$ is a $\mathsf{PaHF}(N; k, 2, t, 2)$. Notably, partitioning hash families are appealing because if a $\mathsf{CA}(N + \rho; v, k, v)$ with ρ constant rows and a $\mathsf{PaHF}(M; \ell, k, t, v)$ exist, then a $\mathsf{CA}(NM + \rho; t, \ell, v)$ exists; this shows that the ingredient CA can be of a different strength than the PaHF. However, PaHFs appear difficult to construct. For recent work on probabilistic methods for PaHFs, see Cassels and Godbole [21].

For some specific t-restrictions, such as covering arrays, one may achieve a smaller number of rows while still satisfying the restriction (e.g., standardizing the CA). Introducing heterogeneity can in some cases provide even more improvements; we content ourselves for now with the homogeneous case. By observing the generality of the framework, much of this survey can be appropriately applied to other types of t-restrictions.

Because a PHF with M rows leads to a TRA with NM rows, one wants the PHF ingredient to have as few rows as possible. The *perfect hash family (row) number*, $\mathsf{PHFN}_\lambda(k, v, t)$, is the minimum N for which a $\mathsf{PHF}_\lambda(N; k, v, t)$ exists. This notation does not extend naturally to heterogeneous hash families, because the number of rows is to be determined. To circumvent this notational issue, we often instead consider maximizing the number of columns rather than minimizing the number of rows. More formally, the *perfect hash family (column) number* $\mathsf{PHHFK}_\lambda(N, \mathbf{v}, t)$ is defined to be the maximum k for which a $\mathsf{PHHF}_\lambda(N; k, \mathbf{v}, t)$ exists. For homogeneous hash families, the notation $\mathsf{PHFK}_\lambda(N, v, t)$ is used. For homogeneous families, one can easily change between row and column numbers:

$$\mathsf{PHFN}_\lambda(k, v, t) = \min(N : \mathsf{PHFK}_\lambda(N, v, t) \geq k)$$

$$\mathsf{PHFK}_\lambda(N, v, t) = \max(k : \mathsf{PHFN}_\lambda(k, v, t) \leq N)$$

A $\mathsf{PHF}_\lambda(N; k, v, t)$ is *optimal* if $N = \mathsf{PHFN}_\lambda(k, v, t)$. Much study has been devoted to determining perfect hash family numbers for as many parameters as possible, as well as what structure underlies optimal PHFs. Moreover, one would hope to provide an explicit representation of the PHF with those parameters, particularly for constructing other combinatorial objects and t-restrictions. If this is not possible, then knowing asymptotics on this quantity is important in helping determine asymptotics for other objects.

2.2 The Basics

First we state elementary relationships among perfect hash family numbers. In order to treat heterogeneous situations as well, we employ perfect hash family column numbers.

Additional rows cannot reduce the number of columns that can be achieved:

Fact 1. $\mathsf{PHHFK}_\lambda(N,(v_1,\ldots,v_N),t) \le \mathsf{PHHFK}_\lambda(N+1,(v_1,\ldots,v_{N+1}),t)$ *whenever* $v_{N+1} \ge 0$.

Reducing the size of column sets to be separated also cannot reduce the number of columns.

Fact 2. $\mathsf{PHHFK}_\lambda(N,(v_1,\ldots,v_N),t) \le \mathsf{PHHFK}_\lambda(N,(v_1,\ldots,v_N),t-1)$ *if* $t \ge 2$.

Reducing λ enables one to remove rows without reducing the number of columns.

Fact 3. $\mathsf{PHHFK}_\lambda(N,(v_1,\ldots,v_N),t) \le \mathsf{PHHFK}_{\lambda-1}(N-1,(v_1,\ldots,v_{i-1},v_{i+1},\ldots v_N),t)$ *if* $\lambda \ge 2$ *and* $1 \le i \le N$.

Increasing the number of symbols in a row cannot reduce the number of columns.

Fact 4. $\mathsf{PHHFK}_\lambda(N,(v_1,\ldots,v_N),t) \le \mathsf{PHHFK}_\lambda(N,(v_1,\ldots,v_{i-1},v_i+1,v_{i+1},\ldots v_N),t)$ *if* $1 \le i \le N$.

Changing the number of columns is also of interest. Removing a column is straightforward, but adding a column can leave $\binom{k}{t-1}$ t-sets of columns unseparated. Naively one could add λ rows for each to obtain

Fact 5. $\mathsf{PHFN}_\lambda(k,v,t) \le \mathsf{PHFN}_\lambda(k+1,v,t) \le \mathsf{PHFN}_\lambda(k,v,t) + \lambda\binom{k}{t-1}$.

Walker and Colbourn [79] show a better bound, later generalized by Martirosyan and van Trung [57]. We improve on their result in Section 3.2.

In order to avoid situations in which a row does not have enough symbols to separate any t-set of columns, we have:

Fact 6. $\mathsf{PHHFK}_\lambda(N,(v_1,\dots,v_N),t) = \mathsf{PHHFK}_\lambda(N-1,(v_1,\dots,v_{i-1},v_{i+1},\dots v_N),t)$ *if* $v_i < t$.

One can also consider reducing the number of symbols in a row,:

Fact 7. $\mathsf{PHHFK}_\lambda(N,(v_1,\dots,v_N),t) \leq \left\lceil \frac{v_i-1}{v_i}\mathsf{PHHFK}_\lambda(N,(v_1,\dots,v_{i-1},v_i-1,v_{i+1},\dots v_N),t)\right\rceil$ *if* $1 \leq i \leq N$.

Iterating Fact 7 until Fact 6 applies, one obtains

Theorem 2.2. (Martirosyan and van Trung [57, Theorem 7.2]) $\mathsf{PHFN}(\lceil\frac{k(t-1)}{v}\rceil, v, t) \leq \mathsf{PHFN}(k,v,t)-1$. *Equivalently,* $\mathsf{PHFK}(N-1,v,t) \geq \lceil \frac{t-1}{v}\mathsf{PHFK}(N, v,t)\rceil$.

Finally we describe a *row amalgamation* method for reducing the number of rows, which essentially comes from [16, 72]. In a $\mathsf{PHHF}_\lambda(N;k,(v_1,\dots,v_N),t)$, select two rows i and j with $1 \leq i < j \leq N$. From these two, form a single row whose entries are ordered pairs, with the first coordinate being the entry from row i and the second from row j. Delete rows i and j (with v_i and v_j symbols), and add the new row with v_iv_j symbols. This method can reduce the number of times a t-set of columns is separated, but this number cannot be reduced to 0.

Fact 8. $\mathsf{PHHFK}_\lambda(N,(v_1,\dots,v_N),t) \leq \mathsf{PHHFK}_{\max(1,\lambda-1)}(N-1,(w_1,\dots,w_{N-1}),t)$ *whenever* $1 \leq i < j \leq N$, $\{v_1,\dots,v_N\}\setminus\{v_i,v_j\} = \{w_1,\dots,w_{N-2}\}$, *and* $w_{N-1} = v_iv_j$.

Now let us dispense with some easier parameter sets. If $N < \lambda$, there are insufficient rows to λ-separate any t-set; so we assume that $N \geq \lambda$. Now $\mathsf{PHFN}_\lambda(k,v,1) = \lambda$

for all k, $v \geq 1$, and $\lambda \geq 1$, because any row separates all 1-sets of columns. Henceforth we only consider cases with $t \geq 2$. Fact 3 underlies the following:

Fact 9. $\mathsf{PHHFK}_\lambda(\lambda, (v_1, \ldots, v_\lambda), t) = \min(v_i : 1 \leq i \leq \lambda)$.

Because of this, we concentrate on cases in which no λ rows are each permitted to contain k or more distinct symbols. It is natural to ask whether one can obtain larger values of k when the number of rows is allowed to exceed λ.

In general, *recursive* constructions combine ingredient PHFs to make 'larger' ones. Many of the facts given provide easy examples of recursive constructions. Of course, because being a perfect hash family of index λ is a t-restriction, so column replacement or composition (Theorem 2.1) is a recursive construction.

Chapter 3

HASH FAMILIES OF HIGHER INDEX

In this chapter, we focus on generating hash families with arbitrary index λ. In Section 3.1, we provide some direct constructions for PHFs of arbitrary index that have a "small" number of rows. In Section 3.2, we take advantage of higher-index ingredients in deriving a new recursive construction, along with a boolean satisfiability formula that aids in the construction; computational results are also provided. In Section 3.3, we analyze PHFN_λ from probabilistic and asymptotic lenses. And finally, in Section 3.4, we use the previous section's results to give a conditional expectation algorithm to construct PHFs of index λ in polynomial time that meet these bounds, along with computational results. Much of this chapter has been submitted in [46].

3.1 Direct Constructions

In many applications, error correction through redundancy in the separation is needed; a few examples are given in [1, 53, 68]. Despite this, there has been little examination of such hash families with $\lambda > 1$, with the notable exception of [3, 4]. [1] We survey a number of the main construction methods for such hash families, with an eye to extending them to treat cases with $\lambda > 1$ when possible. Our emphasis is on fixed values of $\lambda \geq 1$; we only treat cases when λ increases as a function of N in the concluding remarks. We focus on combinatorial aspects, discussing in particular constructive approaches to produce explicit examples for use in applications.

No PHHF_λ with fewer than λ rows exists; when there are λ rows, Fact 9 applies. Suppose that $v_1 \geq \cdots \geq v_N \geq t$ and that a $\mathsf{PHHF}_\lambda(N; v_\lambda + 1, (v_1, \ldots, v_N), t)$ exists.

[1] Also, the only work (as far as we are aware) on covering arrays with regard to higher index was for $\lambda = 2$ in [8].

Using Fact 3 we remove the first $\lambda - 1$ rows to obtain a $\mathsf{PHHF}_1(N - \lambda + 1; v_\lambda + 1, (v_\lambda, \ldots, v_N), t)$. Each row contains at least one pair of columns in which a symbol is repeated. Let $\{\gamma_i, \gamma_i'\}$ be such a pair of column indices with a repeated symbol in row i for $\lambda \leq i \leq N$. Then $\bigcup_{i=\lambda}^{N} \{\gamma_i, \gamma_i'\}$ is a set of at most $2(N - \lambda + 1)$ columns that is separated by no row of the PHHF_1. When $N \leq \frac{t}{2} + \lambda - 1$, this is a contradiction. Hence we conclude

Fact 10. *When* $v_1 \geq \cdots \geq v_N \geq t$*,* $\mathsf{PHHFK}_\lambda(N, (v_1, \ldots, v_N), t) > v_\lambda$ *only if* $N \geq \lceil \frac{t+1}{2} \rceil + \lambda - 1$.

Although this condition is not sufficient whenever $v_1 \geq \cdots \geq v_N \geq t$, it does establish, for example, that $\mathsf{PHFK}_1(N, v, t) = v$ whenever $1 \leq N \leq \frac{t}{2}$. In order to increase the number of columns, one therefore requires further rows.

We describe a $\mathsf{PHF}_\lambda(s + \lambda; m(s + \lambda), m(s + \lambda - 1) + 1, 2s + 1)$ whenever $m \geq 2$ and $s \geq 1$, generalizing a result of Walker and Colbourn [79] when $\lambda = 1$.

Construction 3.1. *Let* $s \geq 1$*,* $m \geq 2$*, and* $\lambda \geq 1$*. A* $\mathsf{PHF}_\lambda(s + \lambda; m(s + \lambda), m(s+\lambda-1)+1, 2s+1)$ *is constructed as follows. Form a set of* $m(s+\lambda-1)$ *elements* X*, and let* ∞ *be an element not in* X*. Then the desired* PHF_λ *contains exactly one occurrence of* ∞ *in each column, and contains each element of* X *exactly once in each row.*

Now Construction 3.1 yields more columns than symbols, and by Fact 10 it has the fewest rows for which this is possible with strength $t = 2s + 1$. As a function of v, k grows linearly. This linear relationship is not restricted to the minimum number of rows, Blackburn [15] explored this phenomenon when $\lambda = 1$, and explicit computations, again for $\lambda = 1$, are pursued in [30]. We apply Blackburn's techniques to treat all λ.

To begin, we suppose that $k > v_1 \geq \cdots \geq v_N$, for otherwise we can either reduce λ by Fact 3 or conclude that one row suffices when $\lambda = 1$. Then every row contains at least one element that is repeated. The key idea is to classify the entries in each row; an entry is a *singleton* for this row when it appears exactly once in the row, and a *replicate* otherwise. Now suppose that a $\mathsf{PHHF}_\lambda(N; k, (v_1, \ldots, v_N), N-(\lambda-2)+s)$ with $0 \leq s \leq \frac{t-1}{2}$ has a column γ with at most $s+\lambda-1$ singletons, and hence at least $N-s-\lambda+1 = t-2s-1$ replicates. Form a set C of $t-2s$ column indices by including γ, and a column index from each of the $t-2s-1$ rows that contains the same symbol as in column γ. Now choose any s further rows, and for each add a pair of column indices for columns containing the same symbol in this row to C. In total, C now contains at most $t-2s+2s=t$ column indices, and C is not separated in any of $t-2s-1+s = N-\lambda+1$ rows. But then C is not λ-separated, because at most $\lambda-1$ rows remain.

So $\lambda+s = t-N+2(\lambda-1)$ is the minimum number of singletons in each column. In a row having v_i symbols, at most v_i-1 can be singletons. However, there must be at least $k(t-N+2(\lambda-1))$ singletons in total. It follows that

$$k(t-N+2(\lambda-1)) \leq \sum_{i=1}^{N}(v_i-1).$$

Hence we obtain

Lemma 3.1. *A* $\mathsf{PHHF}_\lambda(N; k, (v_1, \ldots, v_N), t)$ *with* $N-(\lambda-2) \leq t \leq 2(N-(\lambda-2))-1$ *satisfies*

$$k \leq \max\left(t, v_1, \ldots, v_N, \frac{\sum_{i=1}^{N}(v_i-1)}{t-N+2(\lambda-1)}\right).$$

Lemma 3.1 ensures that for a $\mathsf{PHF}_\lambda(N; k, v, t)$ with $N-(\lambda-2) \leq t \leq 2(N-(\lambda-2))-1$, (or, equivalently, $\frac{t+1}{2}+\lambda-2 \leq N \leq t+\lambda-2$), k grows linearly as a function of v.

For $\lambda = 1$, Blackburn [15] establishes that when $N = t + \lambda - 1$, k grows superlinearly in v. We extend his construction to treat all values of λ next.

Example 3.1. *Let $t \geq 2$, $\lambda \geq 1$, and $a \geq 2$. Then there exists a $\mathsf{PHF}_\lambda(t+\lambda-1; a^{t+\lambda-1}, a^{t-\lambda-2}, t)$. The set of all vectors from $\{1, \ldots, a\}^{t+\lambda-1}$ index the columns. In each column, in the ith row place the vector from $\{1, \ldots, a\}^{t+\lambda-2}$ obtained by deleting the entry in the ith coordinate of the column index.*

The verification that this is a $\mathsf{PHF}_\lambda(t + \lambda - 1; a^{t+\lambda-1}, a^{t-\lambda-2}, t)$ comes essentially from [15]. Suppose to the contrary that there are t rows $\rho_1, \ldots, \rho_t$ in which t columns $\gamma_1, \ldots, \gamma_t$ are not separated. Form a graph G on vertex set $\{\gamma_1, \ldots, \gamma_t\}$; for each $\rho \in \{\rho_1, \ldots, \rho_t\}$, place an edge in G between some two vertices whose columns share a symbol in row ρ, and colour the edge with ρ. Now G has t vertices and t edges (of t different colours), and hence contains a cycle, say on vertices $\{v_0, \ldots, v_\ell\}$. Let $e_i = \{v_i, v_{i+1}\}$ for $0 \leq i < \ell$ have colour c_i, and let $e_\ell = \{v_\ell, v_0\}$ have colour c_ℓ. For $0 \leq i \leq \ell$, the two columns indexed by e_i agree in coordinate c_i *and in no other.* Because all edge colours in the cycle are distinct, the columns indexed by each of $\{e_0, \ldots, e_{\ell-1}\}$ agree in coordinate c_ℓ, but e_ℓ requires that they disagree, which yields the contradiction.

Fact 1 now guarantees that k grows superlinearly in v whenever $N \geq t + \lambda - 1$, in contrast with the requirement that $\mathsf{PHFK}_\lambda(t+\lambda-2, v, t) \leq \max(v, \frac{1}{\lambda}(t+\lambda-2)(v-1))$ from Lemma 3.1.

Of course, practical interest is in obtaining a large number of columns, but understanding the situation with few rows has an important consequence.

Theorem 3.1. *Let $N = \alpha(t - 1) + \beta$ with $1 \leq \beta \leq t - 1$. Then $\mathsf{PHFK}_\lambda(N + \lambda - 1, v, t) \leq \mathsf{PHFK}_1(N, v, t) \leq v^\alpha(t - 1 + \beta(v - 1)) \leq (t - 1)v^{\lceil \frac{N}{t-1} \rceil}$.*

Proof. Consider a $\mathsf{PHF}_1(N; k, v, t)$. Repeatedly amalgamate rows (Fact 8) to form a $\mathsf{PHHF}_1(t-1; k, ((v^{\alpha+1})^{\beta}(v^{\alpha})^{t-1-\beta}), t)$. Apply Lemma 3.1 to conclude that $k \leq v^{\alpha}(t-1+\beta(v-1))$. □

Row amalgamation can reduce λ when it exceeds 1, and hence Theorem 3.1 employs amalgamation only when $\lambda = 1$. Consequently, it yields a useful upper bound when $\lambda = 1$, but we anticipate that the bound is weak when $\lambda > 1$.

In the 'linear' range when $\frac{t+1}{2} + \lambda - 2 \leq N \leq t + \lambda - 2$, Lemma 3.1 establishes that for some constant $c_{N-(\lambda-1),t-N-(\lambda-1),\lambda}$, the existence of a $\mathsf{PHF}_\lambda(N; k, v, t)$ requires that $k \leq c_{N-(\lambda-1),t-N-(\lambda-1),\lambda} v$. Blackburn [15] devises a linear programming formulation to explicitly determine the constant $d_{N,t-N}$ so that $k = d_{N,t-N} v(1 + o(1))$ when $\lambda = 1$. In order to establish the lower bound asymptotically, he develops a construction technique using coverings. In Chapter 4, the method is extended to produce explicit constructions for small values of v, and to treat the generalization to distributing hash families (with $\lambda = 1$).

3.1.1 The Connection with Codes

One direct method for constructing a variety of hash families relies on the existence of error-correcting codes. A *code* with parameters $(n, k, d)_q$ is a set of k distinct vectors (*codewords*) of length n over an alphabet of size q, so that every two distinct codewords are at Hamming distance at least d. Then $A_q(n, d)$ denotes the largest k for which there is a $(n, k, d)_q$ code, and $A(n, d, w)$ denotes the largest k for which there is a $(n, k, d)_2$ code in which each cod word has weight w (i.e., w 1's). See [20] for some bounds on $A(n, d, w)$. Determining exact values for $A_q(n, d)$ and $A(n, d, w)$ in general remains a major challenge.

Alon [2] shows a connection between codes and PHF_1s; see also Atici et al. [10]. We give the easy generalization for higher index:

Theorem 3.2. *If there is an $(n,k,d)_q$ code, then for any t, λ such that $\binom{t}{2} < \frac{n-\lambda+1}{n-d}$, there is a $\mathsf{PHF}_\lambda(n;k,q,t)$.*

Proof. Let C be an $(n,k,d)_q$ code. Construct an array A that has the codewords of C as its columns. Let $L = \{c_1, \cdots, c_t\}$ be a set of t columns of A. Two distinct columns of L can agree in at most $n-d$ rows, so the number of rows in which not all columns of L disagree is at most $(n-d)\binom{t}{2}$. Provided that $(n-d)\binom{t}{2} < n-\lambda+1$, A has at least λ rows that separate L. $\square$

In general, Theorem 3.2 yields a PHF from a code, but not every PHF need arise in this way. However, when $t = 2$ the correspondence is exact (see Mehlhorn [58] and Atici, Magliveras, Stinson, and Wei [10] when $\lambda = 1$):

Theorem 3.3. *An $(n,k,\lambda)_q$ code is equivalent to a $\mathsf{PHF}_\lambda(n;k,q,2)$.*

It follows that $\mathsf{PHFK}_1(N,v,2) = v^N$ and hence $\mathsf{PHFN}_1(k,v,2) = \lceil \frac{\log k}{\log v} \rceil$. By considering all codewords in $\{0,\ldots,v-1\}^N$ whose entries sum to 0 (mod v), one has $\mathsf{PHFK}_2(N,v,2) = v^{N-1}$. When $\lambda \geq 3$, one wants $(n,k,d)_q$ codes with $d \geq 3$. Numerous constructions and bounds are known [56], but in general exact values are not.

There is a $\mathsf{PHHF}_1(N; \prod_{i=1}^{N} v_i, (v_1, \cdots, v_N), 2)$ for any $N \geq 1$ and $v_1, \cdots, v_N \geq 2$, obtained by taking all possible column vectors. In the heterogeneous case, one has a correspondence with codewords in which each coordinate has its own alphabet, but such codes have not been much studied.

Turning to cases with $t \geq 3$, Theorem 3.2 has been extensively employed, particularly to Reed-Solomon codes to make PHFs with $\lambda = 1$ [10]. We formulate a generalization using design-theoretic terminology. A *transversal design*, $\mathsf{TD}(s,k,n)$ is a triple $(V, \mathcal{G}, \mathcal{B})$ where V is a set of kn points, partitioned into k groups $\mathcal{G} = \{G_1, \cdots, G_k\}$, and $|G_i| = n$ for all i. Furthermore, $\mathcal{B}$ contains n^s blocks of size k with

$|B_i \cap G_j| = 1$ for all i, j, and $|B_i \cap B_j| \leq s$ for all $i \neq j$. A standard construction of transversal designs over the finite field $\mathbb{F}_q$ follows.

Construction 3.2. *A* $\mathsf{TD}(s, k, q)$ *with* $k \leq q+1$ *exists. Let* $X = \{x_1, \dots, x_k\} \subseteq \mathbb{F}_q \cup \{\infty\}$*. The elements of the* TD *are* $\mathbb{F}_q \times X$*. For each polynomial* $a_0 + a_1 y + \cdots + a_{s-1} y^{s-1}$ *of degree* $s-1$ *with coefficients from* $\mathbb{F}_q$*, form a block that contains element* (b, z) *whenever* $z \in X$ *and (1)* $b = a_0$ *when* $z = \infty$*, or (2)* $b = a_0 + a_1 z + \cdots + a_{s-1} z^{s-1}$ *otherwise (all arithmetic performed in* $\mathbb{F}_q$*).*

A transversal design constructed in this manner is *linear*. Treating the blocks of the $\mathsf{TD}(s, k, q)$ from Construction 3.2 as columns and the elements of X as rows, we obtain a $k \times q^s$ array C on q symbols. In fact, because the difference between two polynomials of degree at most $s-1$ is also a polynomial of degree at most $s-1$, and such a polynomial has at most $s-1$ roots, the columns of C form a $(k, q^s, k-s)_q$ code, so Theorem 3.2 applies. When constructed in this way, the PHF_λ is *linear* as well. However, because the code has a natural algebraic interpretation, much more can be said. Suppose that there is a set X for which every set t polynomials of degree $s-1$ disagree on some value of X. This can arise when $|X| \leq (s-1)\binom{t}{2}$. For example, Blackburn [14] shows that a $\mathsf{PHF}(3; r^3, r^2, 3)$ exists for all $r \geq 2$, and that a $\mathsf{PHF}(6; p^2, p, 4)$ exists for all primes $p \geq 17$ or $p = 11$. This phenomenon has been extensively examined when $\lambda = 1$. A PHF is *optimal linear* if it is linear and no linear PHF exists having fewer rows. Blackburn [14] provides explicit constructions of PHFs, some of which are optimal.

Blackburn and Wild [18] showed that if q is a sufficiently large prime power, there is an optimal linear $\mathsf{PHF}(s(t-1); q^s, q, t)$ for $s, t \geq 2$. For specific choices of s and t, characterizations of the number of rows that suffice for small prime powers q have been carried out by Barwick and Jackson [11, 12], and Colbourn and Ling [35].

These provide numerous explicit examples of PHF_1s that are easily constructed. The extension of larger values of λ is straightforward.

It remains an open question whether an optimal PHF exists whenever an optimal linear PHF exists [18, 14]. The linear perfect hash families always consist of rows in which the numbers of occurrences of each symbol are as equal as possible. Of course, this equireplication cannot be required for all parameter sets; consider, for example, Construction 3.1. When $s = 1, m = 2$, and $\lambda = 1$, it is possible to prove that the corresponding PHF_2s have a single equivalence class (see Theorem 3.7), and so every row of any $\mathsf{PHF}_1(2; k, v, 3)$ arising from this construction must have this property. Nevertheless, it appears plausible that once the number of rows is large enough, every row can be required to be nearly equireplicated. If true, this constraint could simplify the development of further constructions.

However, we do not expect that linear PHFs lead to the largest number of columns. Consider, for example, $\mathsf{PHFK}(3, v, 3)$. By [14], $\mathsf{PHFK}(3, v, 3) = \Omega(v^{1.5})$; however, Walker and Colbourn [79] found solutions for small v that suggest a larger growth rate, and posed the question of whether $\mathsf{PHFK}(3, v, 3) = o(v^2)$. Fuji-Hara [51] constructs $\mathsf{PHF}(3; v^5, v^3, 3)$ and $\mathsf{PHF}(3; v^2(v + 1), v^2, 3)$ for a prime power $v \geq 3$, to establish that $\mathsf{PHFK}(3, v, 3) = \Omega(v^{5/3})$. Shangguan and Ge [69] solved the question of Walker and Colbourn: For sufficiently large v and arbitrary $\varepsilon > 0$, that $q^{2-\varepsilon} < \mathsf{PHFK}(3, v, 3) = o(v^2)$. A similar result for $\mathsf{PHFK}(4, v, 4)$ is also proved.

One does not need transversal designs constructed over the field $\mathbb{F}_q$ in order to produce a code. It is well known that a $\mathsf{TD}(2, k, v)$ is equivalent to $k - 2$ mutually orthogonal latin squares of side v (see [28], for example). Via this connection, one can generalize a result of Stinson, Wei, and Zhu [76] to $\lambda \geq 1$, by also employing Theorem 3.2:

Theorem 3.4. [76] *If there are at least $s = \binom{t}{2} + \lambda - 2$ MOLS of order n, there exists a* $\mathsf{PHF}_\lambda(s + 2 + \lambda; n^2, n, t)$.

The same authors generalize this statement to mutually orthogonal $n \times m$ latin rectangles on $\max(m, n)$ symbols, obtaining a PHF_1 with mn columns. Dinitz, Ling, and Stinson [44] establish in some cases that the number of rows employed by Theorem 3.4 can be reduced by ensuring that the corresponding TD avoids certain forbidden configurations.

Another generalization of transversal designs is to block designs. Let X be a set of v *points*, and $\mathcal{B}$ be a set of b subsets of X, called *blocks* of X each with k points. Then $(X, \mathcal{B})$ is a *balanced incomplete block design* (BIBD) if every point occurs in r blocks, and every pair of points occurs in λ blocks. We denote this by $\mathsf{BIBD}(v, b, r, k, \lambda)$. By a simple counting argument, $vr = bk$ and $\lambda(v - 1) = r(k - 1)$. A BIBD is a *resolvable balanced incomplete block design* if $\mathcal{B}$ can be partitioned into r *parallel classes*, and each class contains $\frac{v}{k}$ disjoint blocks. We denote this by $\mathsf{RBIBD}(v, b, r, k, \lambda)$. Brickell [19] and Atici, Magliveras, Stinson, and Wei [10] proved that if there is an $\mathsf{RBIBD}(v, b, r, k, \lambda)$ and $r > \lambda\binom{w}{2}$, there is a $\mathsf{PHF}_1(r; v, \frac{v}{k}, w)$. For results on the existence and asymptotics of RBIBDs, see [28].

3.2 A New Recursive Construction for PHFs of Higher Index

In this section, we address the problem of bounding the difference

$$\mathsf{PHFN}_\lambda(ks + d, v, t) - \mathsf{PHFN}_\lambda(k, v, t)$$

for different choices of s, d. Most techniques studied previously either restrict the choices of s, d, limit which values of v, t in which a bound is possible, or the bound itself is not strong.

We provide a framework that does not have any limitations that obtains a bound for every choice of s, d, v, t. However, if one sets $s = 1$ and still lets d be arbitrary, further improvements can be obtained. Even further still, if $s = 1$ and d is a constant, then the best improvements of all can be proved.

An easy (weak) upper bound is that $\mathsf{PHFN}_\lambda(ks + d, v, t) - \mathsf{PHFN}_\lambda(k, v, t) \leq \lambda(\binom{ks+d}{t} - s\binom{k}{t})$, obtained as follows. Horizontally juxtapose the $\mathsf{PHF}_\lambda(N; k, v, t)$ s times, and append any d further columns. Label the columns as $([k] \times \{1, \cdots, s\}) \cup \{\infty_1, \cdots, \infty_d\}$. Then the t-sets which are not separated are those which have two columns having identical first coordinate, or have at least one column among $\{\infty_1, \cdots, \infty_d\}$. There are $\binom{ks+d}{t}$ sets overall, and $\binom{k}{t}$ of them are already separated by assumption for each of the s blocks. It is possible to improve this bound by a factor of 2, because every row can separate any two t-sets. With even more careful analysis based on integer partitions, this same construction can yield a better bound. However, this is not anywhere near the best general bounds that can be obtained.

We review various parameters v, t, s, d that have been investigated previously for bounding $\mathsf{PHFN}_\lambda(ks + d, v, t) - \mathsf{PHFN}_\lambda(k, v, t)$ for general k. In most situations, such bounds are found by appending various columns of the existing PHF, either without modification or with cyclic shifts of the symbols. Walker and Colbourn [79], with improvements by Martirosyan and van Trung [57], considered the cases of (1) $s \geq 2$ constant and $d = 0$, (2) $s = 1$ and d fixed, and (3) $s = 1$ and d arbitrary. We consider (2) and (3) briefly, and return to (1) in the concluding remarks as future work. Their results are inherently limited because it is required that $1 \leq d \leq v - t + 2$, and when $v \approx t$, the number of columns to be added is comparatively small. However, such results are "optimal" in the sense that no other method that appends columns of an existing PHF to itself can improve upon the bounds they achieve. So any method that we produce must improve upon when $d \geq v - t + 1$.

Colbourn and Ling [36] generalize other results by Martirosyan and van Trung to give a very general bound. Their method uses a *difference distributing hash family*, DDHF, as well as the notion of *matroshka type* and *laminar type*. The former is a generalization of a PHF with separation across a partition of the set of t columns based on a given abelian group, and the latter two are tuples indicating a structural property of a hash family (namely how many rows are needed to achieve a PHF or DDHF of strength ℓ for $2 \leq \ell \leq t$). However, their method suffers from (1) very few constructions existing for DDHFs when the number of symbols is small, and (2) the matroshka and laminar types need to be known in advance. In fact, the only constructions for DDHFs require v to be a prime power and $v \geq \binom{t}{2}$, which is less useful if t is large.

We provide a framework for improving upper bounds for $\mathsf{PHFN}_\lambda(ks+d, v, t) - \mathsf{PHFN}_\lambda(k, v, t)$ for the following scenarios: (1) s, d both arbitrary; (2) $s = 1$ and d arbitrary; and (3) $s = 1$ and d a fixed constant. In all cases, there are no requirements on k, s, v, t, or d. When s and k are large, the parameter d is much smaller than $k \times s$. If one has a PHF on $ks + d$ columns, one can find a PHF on ks columns by removing any d of them. Therefore, if a certain number of columns is desired, k', that is not a multiple of k, then one should find the least s such that $ks \geq k'$, and delete columns as necessary.

3.2.1 s Arbitrary

For s and d both arbitrary, we require an additional ingredient. Let t be a positive integer, and $\mathbf{w} = (w_1, \cdots, w_m)$ a nonempty ordered partition of t with $w_1 \leq \cdots \leq w_m$. We say that an $N \times m$ array *fully covers* $\mathbf{w}$ if for every permutation $\rho(\mathbf{w})$ of $\mathbf{w}$, some row of the array is equal to $\rho(\mathbf{w})$. A *partition covering array*, written $\mathsf{PCA}_\lambda(N; t, s)$, is an $N \times s$ array in which for every nonempty ordered partition of size i of $[t]$ for every valid i, every choice of i columns fully covers that partition at least λ times.

We begin with a construction of PCA_λs. For positive integers $r_1, \cdots, r_m$, let $p = \sum_{i=1}^{m} r_i$. A *partitioning ordered design of type* $(w_1, \cdots, w_m)$ *and replication* $(r_1, \cdots, r_m)$, written $\mathsf{POD}_\lambda(N; s, (w_1, \cdots, w_m), (r_1, \cdots, r_m))$, is an $N \times s$ array in which (1) $w_i < w_j$ for all $i < j$, and (2) every p-tuple formed from $(w_1, \cdots, w_m)$ by repeating each w_i exactly r_i times, in any order, is fully covered at least λ times in the POD_λ. A $\mathsf{PCA}_\lambda(N; t, s)$ can then be formed by considering all nonempty partitions of $[t]$, forming the corresponding POD_λ, and vertically juxtaposing all such POD_λs.

We are now ready to state our main construction here. Let $N, k \geq v_1, \cdots, v_N \geq t \geq 2$, and $s \geq 1$ be positive integers, and $1 \leq k_1, \cdots, k_{s-1} \leq k$. Denote $m = k + \sum_{i=1}^{s-1} k_i$. Suppose there exist:

1. a $\mathsf{PCA}_\lambda(N_P; t, s)$ C;

2. a $\mathsf{PHHF}_\chi(N; k, (v_1, \cdots, v_N), t)$ A; and

3. for all $1 \leq i \leq s-1$ and $1 \leq p \leq t-1$, a $\mathsf{PHF}_\alpha(N_{p,i}; k_i, p, p)$ $\mathsf{B}_{p,i}$.

For each $1 \leq i \leq s$, horizontally append any k_i distinct columns of the original PHHF_χ to A. In C, denote $\mathsf{C}_{r,c}$ to be the symbol in row r, column c. Let C_r to be the set of symbol/block pairs in row r of C without multiplicity such that the corresponding number of columns corresponding to r is maximum. By this, we mean that if $\mathsf{C}_{r,c} = \mathsf{C}_{r,c'} = \sigma$ with $c \neq c'$ but $k_c > k_c'$, then C_r contains (σ, c), not (σ, c'). For each row r of C, form the Cartesian product of all $B_{p,i}$ for $p \in \mathsf{C}_r$, taking the minimum resulting number of rows by adding any extra symbols (in the case $v > t$) to any of the corresponding PHFs, in all possible ways; here, PHHF_χs may be formed as a result. (For any symbol/block pair (σ, c) not in C_r, duplicate the corresponding set of columns for σ from some other block, deleting columns as necessary to achieve k_c columns). When the product is formed, vertically juxtapose it to A.

Theorem 3.5. *The array formed from the above construction is a* PHHF *with* m *columns, strength* t*, and index (at least)* $\lambda \cdot \chi \cdot \alpha$.

Proof. Partition the m columns into "blocks" $K, K_1, \cdots, K_{s-1}$, where $|K_i| = k_i$, according to how the columns were appended in the construction. Let T be an arbitrary t-set of columns, and let $\chi = |[k+1, \cdots, m] \cap T|$. If $\chi \leq 1$, then T is already separated since only columns of A were duplicated. If $\chi \geq 2$, then T may not be separated in the first N rows. However, there is some partition of $[t]$ that corresponds to how T is distributed among the blocks. Form the Cartesian product of the corresponding $\mathsf{B}_{p,i}$. Since each ingredient was a PHHF, at least one row in each of the $\mathsf{B}_{p,i}$ separates the required columns; by forming the Cartesian product of these ingredients, some row in the resulting PHHF must separate T. The reasoning for the index being at least $\lambda \cdot \chi \cdot \alpha$ is now immediate. □

Note that there is no correspondence between the blocks of columns chosen in the proof of Theorem 3.5. Let $\mathsf{PODN}_\lambda(s, (w_1, \cdots, w_m), (r_1, \cdots, r_m))$ be the smallest N for which a $\mathsf{POD}_\lambda(N; s, (w_1, \cdots, w_m), (r_1, \cdots, r_m))$ exists. We consider the homogenous case of Theorem 3.5:

Corollary 3.5.1. *Let* $k \geq v \geq t \geq 2$*, and* $s \geq 2$*. Then,*

$$\mathsf{PHFN}_\lambda(ks, v, t) \leq \mathsf{PHFN}_\lambda(k, v, t) + \sum_{\substack{1 \leq w_1 < \cdots < w_m < t \\ r_1, \cdots, r_m \geq 1 \\ r_1 + \cdots + r_m = t \\ m \leq \min(s,t) \\ p, r \geq 1 \\ p \cdot r \geq \lambda}} \mathsf{PODN}_p(s, (w_1, \cdots, w_m), (r_1, \cdots, r_m)) \times f_r,$$

where $f_r = \min_{0 \leq i_1 + \cdots + i_m \leq v-t} \prod_{j=1}^{m} \mathsf{PHFN}_r(k, w_j + i_j, w_j)$.

Proof. Apply Theorem 3.5, by:

1. Setting $k_i = k$ for all $2 \leq i \leq s$;

2. The PCA is formed from the vertical juxtaposition of the corresponding POD ingredients, as outlined previously.

The minimum corresponds to "transferring" symbols between the various PHF ingredients so that at most v symbols are used in every row. $\square$

As a corollary, we recover a theorem of Walker and Colbourn [79]:

Corollary 3.5.2. $\mathsf{PHFN}(2k, 3, 3) \leq \mathsf{PHFN}(k, 3, 3) + 2\mathsf{PHFN}(k, 2, 2)$.

Proof. Apply Corollary 3.5.1; the summation only involves $w_1 = 1, w_2 = 2$, and so only one POD_1 ingredient is needed. A $\mathsf{POD}_1(2; 2, (1, 2), (1, 1))$ can be easily constructed. Since $v = t$, the product is twice that of a PHF of strength 2 with a PHF of strength 1; hence, $2\mathsf{PHFN}(k, 2, 2)$ rows are needed. $\square$

We can generalize Corollary 3.5.2 further:

Corollary 3.5.3. $\mathsf{PHFN}_\lambda(2k, v, t) \leq \mathsf{PHFN}_\lambda(k, v, t) + \sum_{i=1}^{t-1} \min_{\substack{0 \leq d \leq v-t \\ p,r \geq 1 \\ p \cdot r \geq \lambda}} \mathsf{PHFN}_p(k, i+d, i) \cdot \mathsf{PHFN}_r(k, v-d, t-i)$.

Proof. A $\mathsf{POD}(2; 2, (i, t-i), (1, 1))$ can be easily constructed for every i. When $v > t$, extra symbols can be transferred between the two PHFs of strength i and $t-i$. Apply Corollary 3.5.1. $\square$

When $\sum_{i=1}^{m} r_i = t$, $\mathsf{PODN}_\lambda(s, (w_1, \cdots, w_m), (r_1, \cdots, r_m)) \leq \lambda \binom{s}{t} \frac{t!}{r_1! \times \cdots \times r_m!}$, because one can simply cover each possible partition in its own individual row; the proof of Corollary 3.5.2 involves a POD_λ that meets this bound. However, one can do much better than this "worst-case" bound using probabilistic methods.

Let $w_1, \cdots, w_m$ be symbols, $r_1, \cdots, r_m \geq 1$ be integers, and consider an $N \times s$ array, with N determined later, and entries are chosen uniformly and independently at random from $\{w_1, \cdots, w_m\}$. We consider the expected number of column sets of size $t = \sum_{i=1}^{m} r_i$ of partitions such that they are not fully covered in the array. Denote an *interaction* to be the set $\{(c_i, p_i) : 1 \le i \le t\}$, where $c_1, \cdots, c_t$ are distinct columns, and $(p_1, \cdots, p_t)$ is an arbitrary ordered partition, with repetition of symbols as dictated by the r_i values. A POD_λ then fully covers all interactions where the symbols come from ordered partitions.

Let $p = \frac{r_1! \cdots r_m!}{t!}$ be the probability that a fixed interaction is covered, where "covered" indicates that the interaction appears at least λ times. The probability that a fixed interaction does not appear at least λ times in an array with N rows is precisely $\sum_{i=0}^{\lambda} \binom{N}{i} p^i (1-p)^{N-i}$. Therefore, the expected number of uncovered interactions in these N rows is precisely $\binom{s}{t}$ times this probability, since the number of column sets is $\binom{s}{t}$. When this expectation is strictly less than 1, then there exists a POD_λ on these parameters. We give more details about analyzing this bound with respect to PHFs in Section 3.3 and a conditional expectation algorithm to construct the PHFs in Section 3.4, so all of the techniques in these sections can be adapted to PODs as well.

3.2.2 $s = 1$, d Arbitrary

We now focus on when $s = 1$; i.e., there are only two blocks, and one block has fewer than k columns. Much of the reasoning in the following result comes from Corollary 3.5.1.

Theorem 3.6. *Let $x \ge 1$, and let ψ be the minimum i such that for some d with $0 \le d \le v - t$, $\mathsf{PHFN}_\lambda(x+1, i+d, i) = 1$. Then, $\mathsf{PHFN}_\lambda(k+x, v, t) \le \mathsf{PHFN}_\lambda(k, v, t) + \delta + \gamma$, where:*

- $\delta = \mathsf{PHFN}_\lambda(x+1, v, t)$ *if $x \ge \min(\psi, t-1)$, and 0 otherwise; and*
- $\gamma = \sum_{i=2}^{\min(x+1, t-1, \psi)} \min_{\substack{0 \le d \le v-t \\ p, r \ge 1 \\ p \cdot r \ge \lambda}} \mathsf{PHFN}_p(k-1, v-i-d, t-i) \cdot \mathsf{PHFN}_r(x+1, i+d, i)$.

Proof. Append one column x times, and we create the final PHF_λ using the Cartesian product technique as before. If $x \geq t-1$, then add a $\mathsf{PHF}_\lambda(N; x+1, v, t)$ to separate the t-sets that are contained in the added columns (with the entries in these rows in the first $k-1$ columns selected arbitrarily). Once $i = \psi$, then no higher values of i need to be considered, since the corresponding ingredient on the rightmost $x+1$ columns has one row; this implies that the PHF_p ingredient with $x+1$ columns consists of all distinct symbols. Since any $\mathsf{PHF}_p(N; k, v, t)$ is also a $\mathsf{PHF}_p(N; k, v, t-1)$, the theorem statement is proved. □

We recover a theorem of Martirosyan and van Trung [57] from Theorem 3.6, because they use $\psi = 2$ in their result. Even though Theorem 3.5 is very general, Theorem 3.6 indicates that it is not optimized for specific ingredients. Denote a *don't care* position in a PHHF to be one that can be set to any legal value, and the array is still a PHHF. Any part in a partition with size 0 has all of its corresponding positions being don't cares. Without sacrificing many rows, pushing the number of columns past $2k$ seems difficult to count the t-sets in each of the blocks. However, Theorem 3.6 can be improved by using heterogeneous hash families for the ingredients, and a symbol can be moved from one ingredient to the other on a per-row basis instead on a per-ingredient basis.

When x is larger than $v - t + 2$, we improve upon iterative applications of the theorem of Martirosyan and van Trung. Instead of adding a single column x times, we add x columns once. This way, we still separate all of the same t-sets as their method does, but our advantage is that other t-sets are also separated, and were not before. Suppose the x original columns were $c_1, \cdots, c_x$, and the duplicates are $c'_1, \cdots, c'_x$. Then any t-set of columns $C_1, \cdots, C_t$ such that for all $i \neq j$ we have that $C_i \not\equiv C_j \pmod{x}$, then $\{C_1, \cdots, C_t\}$ are separated. Therefore, we only need to separate the t-sets where

$C_i \equiv C_j \pmod{x}$ for some i, j and $1 \leq |\{C_1, \cdots, C_t\} \cap \{c_1, \cdots, c_x\}| \leq t-1$, and similarly for $c'_1, \cdots, c'_x$. There are precisely $\sum_{i=1}^{t-1} \binom{x}{i}\binom{x}{t-i}$ such sets to consider here; however, we desire to find the smallest hash families that only require separation of these t-sets. To do so, we consider a satisfiability formula for PHFs.

3.2.3 A Satisfiability Formula for PHF*s*

Because we only desire to find hash families involving a "small" number of columns and relatively small strength, we turn to satisfiability methods to find small-enough ingredients to assist the results. For a boolean variable x, the two *literals* for x are the *positive form* x, and the *negative form* $\overline{x}$. A boolean formula is in *conjunctive normal form (CNF)* if it is the conjunction (AND, written $\wedge$) of *clauses*, which is a *disjunction*, written $\vee$, of literals. An *assignment* is a function which maps the literals in the formula to $\{0, 1\}$. We say that the formula is *satisfiable* if there is some assignment such that the formula evaluates to 1. We start by building a boolean formula that is satisfiable if and only if a $\mathsf{PHF}_\lambda(N; k, v, t)$ exists, and then show how to improve the representation for the modified problem. Since the hash families considered that will be used in the new recursive construction only require a small number of columns (and hence a small number of rows), we consider a "naive" approach.

We index rows, columns, and symbols starting at 1 for convenience. Let $x_{i,j,s}$ be a boolean variable that indicates that row i for $1 \leq i \leq N$, column j (for $1 \leq j \leq k$) contains value s ($1 \leq s \leq v$). Then we construct the following formula, which is satisfiable if and only if there exists a $\mathsf{PHF}_\lambda(N; k, v, t)$:

$$\left(\bigwedge_{\substack{1\le i\le N\\ 1\le j\le k}} \left(\bigvee_{1\le s\le v} x_{i,j,s} \right) \wedge \left(\bigwedge_{s\neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right) \wedge$$

$$\left(\bigwedge_{1\le c_1<\cdots<c_t\le k} \bigvee_{\substack{S\subseteq[N]\\ |S|=\lambda}} \bigwedge_{R\in S} \bigvee_{\substack{1\le v_1<\cdots<v_t\le v\\ v_i'\in P(v_1,\cdots,v_t)}} (x_{R,c_1,v_1'} \wedge \cdots \wedge x_{R,c_t,v_t'}) \right)$$

where $P(v_1, \cdots, v_t)$ is the set of all permutations of the symbols $v_1, \cdots, v_t$. Intuitively, the first part of the conjunction checks that there is exactly 1 entry in row i and column j, and the second checks that the PHF separates every t-set at least λ times. The separation condition is met by a "witness" set of λ rows for which separation occurs (but of course, the separation in each of the λ rows can be accomplished with any t distinct symbols). We have to use all permutations of the symbols $v_1, \cdots, v_t$ because the entries of the PHF can be in any order.

Note that this formula is not in CNF, because the second part of the conjunction involves a disjunction of conjunctions. Investigations of boolean formula representations of covering arrays have been explored previously [52]; one such idea is the *incidence matrix* representation, which (translated to hash families) is a set of $N \times \binom{k}{t}$ variables $x_{i,C}$ where $x_{i,C} = 1$ if and only if the set of columns C is separated in row i. This problem of expressing when a variable can be set to 1 reduces to that of determining when a set of t-sets $C_1, \cdots, C_m$ can be simultaneously separated in a single row. However, this problem is known to be NP-complete [32], even in the $v = t = 3$ case, via a reduction from the 3-coloring graph problem. Even if it was not NP-complete, the fact that there are so many variables in this representation would make this approach intractable. It remains an open question as to whether or not an efficient (polynomial in size) CNF encoding of whether or not a PHF exists, even when $\lambda = 1$ and k, v, t grow.

We can apply *symmetry breaking* to this formula, which refers to fixing certain variables to be true or false. We do so in the following ways, without loss of generality; each involves appending to the formula above with unit clauses for each of the involved variables, forcing them to be true.

- For all $1 \leq \ell \leq v-1$, the ℓ-th column only can consider symbols from $\{0, \cdots, \ell-1\}$, because we can rename symbols in each row arbitrarily and still maintain the separation property.

- For each row, insist that each value appears at least once in that row. We can enforce this because substituting a value for another that has not appeared yet either keeps the separated t-sets the same, or it separates the same as well as other t-sets (for example, if a symbol that appears at least twice has one of its occurrences replaced by a new value).

- For the first λ rows, separate the first t columns λ times, and require that the other $N-\lambda$ rows do not, because if we desire to find a PHF that is optimal, then at least one t-set is separated exactly once (again without loss of generality, we can assume this t-set has this property).

We can also apply the following symmetry breaking strategy, but for only for when the number of rows is "large enough": enforce that in the first row, column i has value $i \pmod v$. In other words, the first row cycles through the symbol set. This separates the largest possible number of t-sets for a single row in isolation, but it is not possible to generate an optimal PHF with a row having this form for all parameter situations, which we prove next.

Construction 3.1 provides a $\mathsf{PHF}_\lambda(s+\lambda; m(s+\lambda), m(s+\lambda-1)+1, 2s+1)$. One may ask as to whether there is a $\mathsf{PHF}_\lambda(s+\lambda-c; m(s+\lambda), m(s+\lambda-1)+1, 2s+1)$ for some $c \geq 0$ and $\lambda \geq 1$; it is certainly true when $c=0$, and it is false for $c \geq 1$ and

$\lambda = 1$ by Walker and Colbourn [79]. However, it cannot be true in all other cases; as an example, set $\lambda = 2$ and consider the case of a $\mathsf{PHF}_1(2; 8, 5, 3)$, which exists by their result. Clearly, $\mathsf{PHFN}_2(8, 5, 3) \leq 4$; we show that it cannot equal 3.

Theorem 3.7. $\mathsf{PHFN}_2(8, 5, 3) = 4$.

Proof. In the proof of Construction 3.1 for when $\lambda = 1$, every 3-set is 1-separated, and none are at least 2-separated. Also note that the first row of the $\mathsf{PHF}_1(2; 8, 5, 3)$ is 12345555, and the second row is the reversal of the first. We prove the statement by subdividing it into three cases for when the number of 5s in this first row, $\#_5$, is in $\{2, 3, 4\}$. We do not need to consider the case of $\#_5 = 0$ because disallowing values from a row cannot help in separating t-sets, nor the cases of $\#_5 = 1$ or $\#_5 \geq 5$ because they are each equivalent to another case (by permuting symbols as needed).

Suppose $\#_5 = 2$; without loss of generality, the row is 54321543 (by permuting columns). The unseparated sets are, after removing set notation for clarity: 015, 016, 025, 027, 035, 045, 056, 057, 126, 127, 136, 146, 156, 167, 237, 247, 257, and 267 (indexed columns starting at 0). We now attempt to separate all of these 3-sets in one row; if we cannot, then three rows are required for every 3-set to be separated once. But to achieve $\lambda = 2$, we must need an additional row after such a third row; therefore, it suffices to prove that all of these 3-sets cannot be simultaneously separated.

We build the second row, initially all indeterminates: $[x; x; x; x; x; x; x; x]$. At each point, either we will fix a coordinate in this row to a value, or maintain a list of candidate values that can be assigned to that index (with square brackets). We can separate 015 arbitrarily, so suppose it is separated by values 1, 2, 3; the row is $[1; 2; x; x; x; 3; x; x]$. Then 016 allows index 6 to be values 3, 4, or 5; but 056 forces the value 3 to be removed from index 6: $[1; 2; x; x; x; 3; [4, 5]; x]$. 025 allows for the 2nd index to be 2, 4, or 5; but 126 removes the 2: $[1; 2; [4, 5]; x; x; 3; [4, 5]; x]$. 027 allows for the 7th index to be 2,3,4, or 5; but 127 removes the 2: $[1; 2; [4, 5]; x; x; 3; [4, 5]; [3, 4, 5]]$. 035 allows for the 3rd index to be 2,4,5, but 136 removes the 2: $[1; 2; [4, 5]; [4, 5]; x; 3; [4, 5]; [3, 4, 5]]$. 045 is the same as the 3rd index: $[1; 2; [4, 5]; [4, 5]; [4, 5]; 3; [4, 5]; [3, 4, 5]]$. Because of 267, this forces the 7th index to be a 3 (since any other choice must conflict with either index 2 or 6): $[1; 2; [4, 5]; [4, 5]; [4, 5]; 3; [4, 5]; 3]$. But then 257 cannot be separated, since only two values (i.e., 4 and 5) are possible in those three coordinates. The proofs for $\#_5 \in \{3, 4\}$ are very similar. □

Suppose that a PHF_1 is being built one row at a time; start the first row containing every symbol as equally often as possible. Then this row separates the maximum possible number of t-sets. Because of the $\mathsf{PHF}_1(2; 8, 5, 3)$ example above, one cannot assume in general that if $\mathsf{PHFN}_1(k, v, t) = N$, then there exists a $\mathsf{PHF}_1(N; k, v, t)$ with one of its rows having this property. Through extensive search, it appears that when $N \geq \lceil \frac{t+1}{2} \rceil$, this assumption can be made; and when the conditions of Construction 3.1 are met, it cannot be. However, proving what are necessary and sufficient conditions for separability is an open problem. Furthermore, one cannot conclude that if a PHF has every t-set separated exactly once that the only solution to achieve a given index λ is to vertically juxtapose it λ times; understanding when this is the case is also an open problem.

3.2.4 Improving PHFN_λ *with Heterogeneous Ingredients*

We state improvements to PHFN_λ by adding x columns once instead of one column x times. Note that when $x = 1$, we recover the Martirosyan and van Trung result above, so consider when $x \geq 2$. Let a $\mathsf{PHF}_\lambda(N; (k_1, k_2), v, t)$ be an $N \times (k_1 + k_2)$ array on v symbols (index columns by $\{1, \cdots, k_1 + k_2\}$) where only the t-sets of columns C with the properties (1) $|C \cap \{1, \cdots, k_1\}| \geq 1$ and (2) $|C \cap \{k_1 + 1, \cdots, k_1 + k_2\}| \geq 1$

are required to be λ-separated. When a t-set C has this property, we say that it *crosses the partition*. The most common case we will consider is when $k_1 = k_2$. Also define a $\mathsf{PHHF}_\lambda(N; (k_1, k_2), (v_1, \cdots, v_N), t)$ similarly. Let $\mathsf{PHFN}_\lambda((k_1, k_2), v, t)$ be the smallest N for which a $\mathsf{PHF}_\lambda(N; (k_1, k_2), v, t)$ exists. We highlight an example of how this object can yield improvements over Theorem 3.6.

Theorem 3.8. *Let ψ be as in Theorem 3.6. Then for all $x \leq t-1$,* $\mathsf{PHFN}(k+x, v, t) \leq \mathsf{PHFN}(k, v, t) + \sum_{i=2}^{\min(x,\psi)} \min_{0 \leq d \leq v-t} \mathsf{PHFN}(k-x, v-i-d, t-i) \cdot \mathsf{PHFN}((x,x), i+d, i)$*.*

Note that $\mathsf{PHFN}((k_1, k_2), v, 2) = 1$ for any k_1, k_2 and $v \geq 2$, because the first k_1 columns can be set to one symbol, and the other k_2 columns set to a different symbol; every 2-set crossing the partition is separated. However, it is simple to derive an example where $\mathsf{PHFN}((k_1, k_2), v, 3) \geq 2$. In fact, these objects exhibit logarithmic growth, just like PHFs, if we apply standard probabilistic techniques.

Here, we can apply even more symmetry breaking to our SAT formula in the case of heterogeneous hash families that contain a partition:

- If the PHHF desired has symbols $(v_1, \cdots, v_N)$ for its rows, then insist that row i only have symbols from $\{1, \cdots, v_i\}$ for all i.
- Further insist that for each row i, columns $1, \cdots, k_1$ can contain symbols only from $\{1, \cdots, t-1\}$, and that columns $k_1+1, \cdots, k_1+k_2$ contain symbols only from $\{v_i - t + 1, \cdots, v_i\}$. This last condition restricts the dimension of the search space since no t-set needs to be separated entirely on one side of the partition.

Results for the existence of PHHFs with 4 rows, a small number of columns, at most 5 symbols for each row, $t = 3$, and $\lambda = 3$ appear in Tables 3.1 and 3.2; a ✓ indicates that the PHHF was found, and an ✗ indicates that no PHHF of those

Symbols ↓, k →	4	5	6	7	8	9	10	11
(3,3,3,3)	✓	✗						
(4,3,3,3)	✓	✗						
(4,4,3,3)		✓	✗					
(4,4,4,3)			✓	✗				
(4,4,4,4)					✓	✗		
(5,3,3,3)		✓	✗					
(5,4,3,3)			✓	✗				
(5,4,4,3)			✓	✗				
(5,4,4,4)					✓	✗		
(5,5,3,3)			✓	✗				
(5,5,4,3)			✓	✗				
(5,5,4,4)					✓	✗		
(5,5,5,3)				✓	✗			
(5,5,5,4)						✓	?	
(5,5,5,5)							✓	?

Table 3.1: Existence of PHHFs with 4 Rows, at Most 11 Columns, at Most 5 Symbols for Each Row, Strength 3, and Index 2.

parameters exists; a question mark indicates that a timeout of 1 day of solving time was reached without proving (un)satisfiability. Most of the entries were solved within a few seconds, whereas some of the ✗ entries took several hours. Entries to the left of a ✓ entry and to the right of an ✗ entry are left blank because deleting columns does not make the formula unsatisfiable when it was satisfiable previously, and adding columns does not allow the formula to become satisfiable when it was unsatisfiable previously. We predict that the two ? entries in Table 3.1 will be ✗, given the length of solving time needed.

Symbols ↓, k →	3	4	5	6
(3,3,3,3)	✓	✗		
(4,3,3,3)	✓	✗		
(4,4,3,3)		✓	✗	
(4,4,4,3)		✓	✗	
(4,4,4,4)		✓	✗	
(5,3,3,3)	✓	✗		
(5,4,3,3)		✓	✗	
(5,4,4,3)		✓	✗	
(5,4,4,4)		✓	✗	
(5,5,3,3)		✓	✗	
(5,5,4,3)			✓	✗
(5,5,4,4)			✓	✗
(5,5,5,3)			✓	✗
(5,5,5,4)			✓	✗
(5,5,5,5)			✓	✗

Table 3.2: Existence of PHHFs with 4 Rows, at Most 6 Columns, at Most 5 Symbols for Each Row, Strength 3, and Index 3.

We construct optimal PHHFs with "small" parameters because they improve on the recursive construction even further; we illustrate this with an example. If the heterogeneous hash family's symbols are, for example, of the form $(i+d)^j(i+d-1)^{N-j}$ (i.e., j of the rows contain at most $i+d$ symbols, and the others contain at most $i+d-1$ symbols), then two PHFs can be used as the other ingredient; one with $v-i-d$ symbols, and the other with $v-i-d+1$ symbols. If only homogeneous hash families were used for both ingredients, then only one pairing of symbols is possible, whereas more can occur here.

Let C be a t-set of columns, and let χ be the cardinality of the intersection of C with the original x columns and their duplicates. Throughout, if x columns are added, then when χ is maximum, if possible, we use an ingredient consisting of all different symbols (if not, we use our SAT solver to find the smallest ingredient possible). Throughout, when we "use" an ingredient, we assume that the PHF ingredient that will be crossed with the secondary one does not contain the symbols given. By convention, the quantity $\mathsf{PHFN}(k, v, t)$ for when $t \leq 1$ is 1; in fact, if some secondary ingredient of strength t consists of all different symbols, then higher values of t do not need to be considered. For each, if we increase the number of columns by x, we assume that the original PHF ingredient had at least x columns. There are possible improvements to these results when $v > t$, because a symbol can be moved from the first ingredient to the second. All of the ingredients presented here are optimal with respect to the number of rows. Of course, the bound on t is not absolutely necessary; for example, if $t = 3$ for Lemma 3.2, then the $\chi = 4$ ingredient is not needed. Only when t is at least than the stated bound are all of the ingredients necessary. As was done in Theorem 3.6, depending on the number of columns that are added, the number of symbols desired, and the strength, some of the constructions may improve on the results stated below by removing unnecessary ingredients.

Lemma 3.2. $\mathsf{PHFN}(k+2, v, t) \leq \mathsf{PHFN}(k, v, t) + \mathsf{PHFN}(k-2, v-2, t-2) + 2\mathsf{PHFN}(k-2, v-3, t-3) + \mathsf{PHFN}(k-2, v-4, t-4)$.

Proof. For $\chi = 2$, a $\mathsf{PHF}(1; (2, 2), 2, 2)$ can be constructed by hand. For $\chi = 3$, use the $\mathsf{PHF}(2; (2, 2), 3, 3)$ described by the two rows 1132 and 1233. (By this, 1132 is the first row, and 1233 is the second.) It is routine to check that this is a $\mathsf{PHF}(2; (2, 2), 3, 3)$ and that there is no $\mathsf{PHF}(1; (2, 2), 3, 3)$. For $\chi = 4$, use $\mathsf{PHF}(1; (2, 2), 4, 4)$ described by the single row 1234. Apply Theorem 3.8. □

As a corollary, we obtain that $\mathsf{PHFN}(k+2,4,4) \leq \mathsf{PHFN}(k,4,4) + \lceil \frac{\log(k-2)}{\log(2)} \rceil + 2$. There is a $\mathsf{PHF}(7;25,5,4)$ [36] and a $\mathsf{PHF}(3;5,4,4)$ [79]; their composition produces a $\mathsf{PHF}(21;25,4,4)$, which currently is the best-known size of a PHF with 25 columns, 4 symbols, and strength 4. The previous best-known upper bound for $k = 27, v = t = 4$ is $\mathsf{PHFN}(27,4,4) \leq 31$; this result improves the upper bound to 28. We continue with additional results; we only outline when more symbols can be useful for the homogeneous case in the following lemma, even though more symbols can be useful in other results.

Lemma 3.3. *Suppose $v = t$. Then,* $\mathsf{PHFN}(k+3,v,t) \leq \mathsf{PHFN}(k,v,t) + \mathsf{PHFN}(k-3, v-2,t-2) + 2\mathsf{PHFN}(k-3,v-3,t-3) + 5\mathsf{PHFN}(k-3,v-4,t-4) + 3\mathsf{PHFN}(k-3, v-5,t-5) + \mathsf{PHFN}(k-3,v-6,t-6)$.

Suppose $v = t+1$. Then, $\mathsf{PHFN}(k+3,v,t) \leq \mathsf{PHFN}(k,v,t) + \mathsf{PHFN}(k-3,v-2, t-2) + 2\mathsf{PHFN}(k-3,v-3,t-3) + \min\{f,g\} + \mathsf{PHFN}(k-3,v-5,t-5)$*, where:*

- $f = 5\mathsf{PHFN}(k-3,v-4,t-4)$*, and*
- $g = 2\mathsf{PHFN}(k-3,v-5,t-4) + \mathsf{PHFN}(k-3,v-4,t-4)$.

Proof. The ingredients are as follows:

- $v = t$: For $\chi = 3$, use 121332 and 112323. For $\chi = 4$, use 123244, 113442, 123423, 111234, and 121343. For $\chi = 5$, use 123445, 123453, and 112435. For $\chi = 6$, use 123456.
- $v = t+1$: For $\chi = 3$, use the corresponding $v = t$ ingredient. For $\chi = 4$, use the following $\mathsf{PHHF}(3;(3,3),(5,4,4),4)$: 112345, 123443, and 123434; or use the corresponding ingredient from the $v = t$ case if it produces fewer rows. For $\chi = 5$, use 123456.

In any of the cases, apply Theorem 3.8. □

We do not enumerate more cases of when $v \geq t+2$, mainly for simplicity of presentation; but one can use the results in, for example, Tables 3.1 and 3.2 to enumerate the several minimums taken over various PHFN results to improve them when the number of symbols increases. We describe two of the many possible improvements from the existing PHFN tables [45] for when the number of columns is moderately large compared to the strength.

Corollary 3.8.1. $\mathsf{PHFN}(364, 6, 6) \leq 1871$ *(Previous best-known: ≤ 2011).*

Proof. There exist a $\mathsf{PHF}(1648; 361, 6, 6)$, a $\mathsf{PHF}(120; 358, 4, 4)$, a $\mathsf{PHF}(27; 358, 3, 3)$, a $\mathsf{PHF}(9; 358, 2, 2)$, and a $\mathsf{PHF}(1; 358, 1, 1)$. Apply Lemma 3.3 $\square$

We analyze the proof of Lemma 3.3: it turns out that $2\mathsf{PHFN}(k-3, v-5, t-4) + \mathsf{PHFN}(k-3, v-4, t-4)$ is smaller than $5\mathsf{PHFN}(k-3, v-4, t-4)$ for the case of $v = 7, t = 6$, mainly because it will yield an improvement if and only if $\mathsf{PHFN}(k-3, v-5, t-4) < 2\mathsf{PHFN}(k-3, v-4, t-4)$. Because $t = 6$, this is equivalent to $\log(v-4) < \log(v-5)$, which is always true when $v \geq 7$. Therefore, we can improve the statement of the result for this case as follows:

Lemma 3.4. $\mathsf{PHFN}(k+3, 7, 6) \leq \mathsf{PHFN}(k, 7, 6) + \mathsf{PHFN}(k-3, 5, 4) + 2\mathsf{PHFN}(k-3, 4, 3) + 2\lceil \log_2(k-3) \rceil + \lceil \log_3(k-3) \rceil + 1.$

Corollary 3.8.2. $\mathsf{PHFN}(364, 7, 6) \leq 774$. *(Previous best-known: ≤ 795)*

Proof. There exists a $\mathsf{PHF}(672; 361, 7, 6)$, a $\mathsf{PHF}(41; 358, 5, 4)$, a $\mathsf{PHF}(18; 358, 4, 3)$, a $\mathsf{PHF}(6; 358, 3, 2)$, a $\mathsf{PHF}(9; 358, 2, 2)$, and a $\mathsf{PHF}(1; 358, 1, 1)$. Apply Lemma 3.4. $\square$

For all of our results, we employed `limboole` [2] to convert our PHHF encoding into a formula in conjunctive normal form, which was then fed as input into the `glucose` satisfiability solver, based on `minisat` [48].

[2] Available at `http://fmv.jku.at/limboole/`.

3.3 Probabilistic and Asymptotic Methods

We first employ the probabilistic method to obtain upper bounds on PHFN_λ. Let A be a random $N \times k$ array over v symbols; by this, each entry is uniformly selected at random from a v-set, independent of all other entries. Choose a t-set of columns $C = \{c_1, \cdots, c_t\}$; the probability that C is not separated in A is the probability that no rows among $c_1, \cdots, c_t$ have all distinct entries. The probability that a given row of size t is all distinct, with v values for each entry, is $p = \frac{\binom{v}{t} t!}{v^t}$. Therefore, the probability that a given row does not separate C is $1 - p$, and the probability that all rows does not separate C is $(1-p)^N$. By linearity of expectation, the expected number of t-sets of columns that are not separated in A is $\binom{k}{t}(1-p)^N$. Solving for N when this expectation is strictly less than 1 guarantees that there must be some PHF_1 with those parameters. Taking logarithms, and upper bounds, we have the following, first proved by Mehlhorn [58]. Throughout, let $f(v,t) = \log(v^t) - \log(v^t - t!\binom{v}{t})$.

Theorem 3.9. $\mathsf{PHFN}_1(k, v, t) \leq \left\lceil \frac{\log \binom{k}{t}}{f(v,t)} \right\rceil$.

We can improve these bounds using the Lovász local lemma, a tool extensively used in combinatorics [7]. We mainly employ the symmetric version here:

Theorem 3.10. *[7] Let $E_1, \cdots, E_n$ be events in a probability space, E_i is mutually dependent of at most d other events for all i, and $\Pr[E_i] \leq p$ for all i. If $ep(d+1) \leq 1$, then with nonzero probability all of $E_1, \cdots, E_n$ simultaneously do not occur.*

In the setting of PHFs, the events E_i are that the ith t-set of columns is not separated. We have that $\Pr[E_i] = (1-p)^N$ from above. Two different events E_i, E_j are mutually dependent if and only if the corresponding t-sets have nonempty intersection, implying that $d = \binom{k}{t} - \binom{k-t}{t} - 1$. So if $e\,(1-p)^N \left(\binom{k}{t} - \binom{k-t}{t}\right) \leq 1$, then there is a PHF with those parameters, proved by Deng, Stinson, and Wei [43].

Theorem 3.11. *[43]* $\mathsf{PHFN}_1(k,v,t) \le \left\lceil \frac{\log\left(\binom{k}{t}-\binom{k-t}{t}\right)+1}{f(v,t)} \right\rceil$.

Stinson, van Trung, and Wei [75] improved this bound further, with the so-called "expurgation" method (which we generalize here using an idea from [34]); this is sometimes called "oversampling," or even "post-processing" [78]. This method was the best known probabilistic bound for PHFN_1 in the general case until an improvement in many parameter sets was found by Procacci and Sanchis [63] using the algorithmic cluster expansion local lemma. Their techniques involved more carefully analyzing the Moser-Tardös algorithm [59], which involves resampling "bad" events (i.e., sets of t columns not being separated). Their approach can be generalized for higher-index PHFs, since all that changes is the probability of each bad event occurring. For simplicity of presentation, we state the post-processing result:

Theorem 3.12. $\mathsf{PHFN}_1(k,v,t) \le \min_{x\ge 0} \left\lceil \frac{\log\binom{k+x}{t}-\log(x+1)}{f(v,t)} \right\rceil$.

Proof. Start with an array with $k+x$ columns, and let E be the expected number of unseparated t-sets of columns in this array if each entry is chosen uniformly and independently at random. If $E < x+1$, then by deleting one column from each of the (at most) x unseparated t-sets, we must have a $\mathsf{PHF}_1(N;k,v,t)$. The theorem statement then is equivalent to solving the following equation for N: $\binom{k+x}{t}(1-p)^N < x+1$, and finding the minimum over all $x \ge 0$. □

In [34] in the context of CPHFs, the value $x = \frac{k}{t-1}$ was chosen and shown to improve on basic probabilistic arguments; the same value works for PHFs as well. Understanding the optimal choice of x remains an open problem. However, all of the above results are for when $\lambda = 1$; we now provide an upper bound for arbitrary λ.

Theorem 3.13. $\mathsf{PHFN}_\lambda(k,v,t)$ *is at most* $c_1 \log k + c_2\sqrt{\lambda \log k}$ *for constants* c_1, c_2 *that only depend on* v,t.

Proof. We combine oversampling with the basic probabilistic method to obtain the theorem statement. Let A be a random $N \times k$ matrix, p be as before, $C = \{c_1, \cdots, c_t\}$ be t columns of A, and X_C be the random variable that is the number of rows in which C is separated in A. The probability that C is separated for a given row is p. We can determine that $\Pr[X_C = i] = \binom{N}{i}(1-p)^{N-i}p^i$. So $\Pr[X_c \leq \lambda - 1] = \sum_{i=0}^{\lambda-1} \Pr[X_C = i] = \sum_{i=0}^{\lambda-1} \binom{N}{i}(1-p)^{N-i}p^i$. Therefore, when

$$\binom{k+x}{t} \sum_{i=0}^{\lambda-1} \binom{N}{i}(1-p)^{N-i}p^i < x+1 \tag{3.1}$$

for some $x \geq 0$, there is a $\mathsf{PHF}_\lambda(N; k, v, t)$.

We can see that $E[X_C] = Np = \mu$. Therefore, we can write the previous probability as $\Pr[X_C \leq (1-\delta)\mu]$ where $\delta = 1 - \ell/\mu$ and $\ell = \lambda - 1$. Note that $0 \leq \delta \leq 1$. So, we can apply Chernoff bounds [7] to obtain that this probability is at most $\exp(-\delta^2\mu/2) = \exp(-\mu/2 \times (1-\ell/\mu)^2)$. This is equal to $\exp(-\mu/2(1-2\ell/\mu+\ell^2/\mu^2)) = \exp(-\mu/2 + \ell - \ell^2/(2\mu))$.

Substituting back, we obtain $\exp(-Np/2 + \lambda - 1 - (\lambda-1)^2/(2Np))$. So if $\binom{k+x}{t} \exp(-Np/2 + (\lambda-1) - (\lambda-1)^2/(2Np)) < x+1$, then there is a $\mathsf{PHF}_\lambda(N; k, v, t)$. Note that if k, v, t, x, λ are fixed, and N increases, then the left-side of this expression is monotonically decreasing. Therefore, we instead solve the above expression for equality, then add 1 to N afterwards.

This is equivalent to $\exp(Np/2 - (\lambda-1) + (\lambda-1)^2/(2Np)) = \binom{k+x}{t}/(x+1)$. Taking logarithms gives: $Np/2 - (\lambda-1) + (\lambda-1)^2/(2Np) = \log\binom{k+x}{t} - \log(x+1)$. Forming the statement in terms of a quadratic formula to solve in terms of N, we find that $N = \frac{1}{p}\left(B + (\lambda-1) + \sqrt{B(2(\lambda-1)+B)}\right) + 1$, where $B = \log\binom{k+x}{t} - \log(x+1)$. So we can see that $\mathsf{PHFN}_\lambda(k, v, t)$ is $c_1 \log k + c_2\sqrt{\lambda \log k}$ for appropriate constants c_1, c_2, and by setting $x = 0$, for example. □

One can interpret Equation (3.1) in that at most x sets of columns are not λ-

separated (or more), and removing one column from each set guarantees the existence of a PHF_λ. If one instead considers an analogous inequality:

$$\binom{k}{t}\sum_{i=0}^{\lambda-1}\binom{N}{i}(1-p)^{N-i}p^i < x+1, \tag{3.2}$$

then this means that again at most x sets of columns are not λ-separated. It may be possible that all x of these sets are not separated at all. However, we can "complete" this array by adding λ rows for each of these sets. Therefore, if N satisfies Equation (3.2), then $N + x\lambda$ rows suffice to create a PHF_λ. This is not as powerful as the result in Theorem 3.13. However, consider any set C that is exactly i-separated; then only $\lambda - i$ rows are needed. So, it suffices to compute the N for which the following inequality is satisfied, and take the minimum over all x:

$$\binom{k}{t}\sum_{i=0}^{\lambda-1}(\lambda-i)\binom{N}{i}(1-p)^{N-i}p^i < x+1. \tag{3.3}$$

Sarkar and Colbourn [67] use a *discrete* probabilistic approach in generating covering arrays, in that they construct the array one row-at-a-time, such that each row is at least as good as the average at that point (by this, each row covers a number of interactions at least the average among all possible rows). In their construction, at each iteration, phrased in the language of hash families, the number of unseparated t-sets is always an integer; therefore, the number of rows can be significantly less than what is determined by, say, Theorem 3.9.

Their approach appears difficult to analyze with hash families of higher index because one now needs to keep track of λ variables; suppose the variables are $A_0, \cdots,$ $A_{\lambda-1}$, where A_i represents the current expected number of t-sets that are separated exactly i times. Initially, $A_0 = \binom{k}{t}$, and $A_i = 0$ for all other i. Consider variables A_0 and A_1. We can guarantee that A_0 will never increase, but A_1 may possibly when an added row separates many of the t-sets corresponding to A_0, and not many corresponding to A_1. We return to this point in Chapter 6.

We consider when N becomes large and analyze the bound asymptotically, instead of what a bound on N there is for *every* choice of k. We suppose that v, t, λ are fixed; then the probability p of separation in a row is also fixed. Consider the failure probability for a given t-set of columns: $\sum_{i=0}^{\lambda-1} \binom{N}{i} p^i (1-p)^{N-i}$. Since p (and consequently $1-p$) is fixed, then this is asymptotically at most $N^\lambda(1-p)^N$. So if $\binom{k}{t} N^\lambda (1-p)^N < 1$ for a choice of N, then the true value of $\mathsf{PHFN}_\lambda(k, v, t)$ is asymptotically at most N as k becomes large.

Without loss of generality, we solve the equation $\binom{k}{t} N^\lambda (1-p)^N = 1$ for N, because any larger value of N would make this a strict inequality; we can do this because we are only observing the bound asymptotically. The *Lambert W-function* is the inverse of the function $f(W) = W \exp(W)$.

Lemma 3.5. *When x is large, $W(x) \approx \log x - \log\log x + o(1)$, where $o(1) \to 0$ as $x \to \infty$.*

Proof. An equivalent definition of $W(x)$ is that it satisfies $W(x) = \log(x/W(x))$. So $W(x) = \log(x/\log(x/\log(x/\cdots)))$. Expansion of the right-hand side yields that this is at most $\log x - \log\log x + o(1)$ when x is large. $\square$

However, since λ is fixed, the bound that results is a constant times $\log k$, which is expected. Nevertheless, the W-function is still useful when λ grows, which is what we prove next.

Theorem 3.14. *Suppose k and λ are large, but λ grows asymptotically slower than $\log k$.* [3] *Furthermore, suppose v, t are fixed. Then $\mathsf{PHFN}_\lambda(k, v, t)$ grows at most $\log k + \lambda \log\log k + o(\lambda)$.*

[3]All we need is that $\lambda < N/2$ for a given fixed N, where N is the number of rows, so λ being asymptotically slower than $\log k$ guarantees this since N will always be at least $\log k$.

Proof. Observe the failure probability for PHFs as outlined before, which is $\sum_{i=0}^{\lambda-1} \binom{N}{i} p^i (1-p)^{N-i}$. Since λ is not fixed, we provide an upper bound for this sum as follows. Instead of giving asymptotics for the whole sum, we upper bound each of the three terms inside the sum, and multiply this product by λ (the number of terms in the sum). Since λ grows slower than $\log k$, we obtain an upper bound of $\lambda\binom{N}{\lambda}(1-p)^{N-\lambda}$. By Stirling's approximation, this is asymptotically equal to $f(p, N, \lambda) = \lambda \frac{N^\lambda}{\lambda!}(1-p)^{N-\lambda}$.

We desire to know what value of N satisfies $\binom{k}{t} f(p, N, \lambda) < 1$. But since one can solve for N when $\binom{k}{t} f(p, N, \lambda) = 1$, and only concern ourselves with asymptotics, we can now use the $W(x)$ function as defined before. The value that N satisfies the above equation asymptotically yields (after removing constant terms inside the W function):

$$N = \frac{\lambda W\left((\lambda!/\binom{k}{t})^{1/\lambda}\right)}{\log(1-p)}.$$

By Lemma 3.5, the numerator is asymptotically equal to:

$$\lambda \log \lambda - \log k - \lambda \log\log k - o(\lambda),$$

since $\log \lambda! \approx \lambda \log \lambda$. The fact that $0 < p < 1$ implies that $\log(1-p) < 0$; multiplying by -1 because of this gives the original theorem statement (since λ grows slower than $\log k$). $\square$

If λ is $o\left(\frac{\log k}{(\log\log k)^2}\right)$, then Theorem 3.14 improves upon Theorem 3.13; this can be proved by solving for λ in the equation $\log k + \lambda \log\log k < \log k + \sqrt{\lambda \log k}$. This is a reasonable choice for the growth of λ, as Theorem 3.14 has λ being $o(\log k)$.

The bounds we derived here are $\log k + \lambda \log\log k + o(\lambda)$ for the upper bound, and the simple bound of $\log k + \lambda$ for the lower one, which comes directly from Fact 10. Even though $\log\log k$ is quite small compared to k, it is still not a constant, and it would be of great interest in removing it if possible. One technique would be finding a better asymptotic analysis on the failure probability, as the rest of the proof did not use any result that is not asymptotically equal to what was derived; in other words, if any improvement is to be made, its proof must improve on an asymptotic formulation for the failure probability.

3.4 A Conditional Expectation Approach

Even though the probabilistic bounds obtained in the previous section have a large gap between the lower and upper bound, we describe a new conditional expectation-type approach to produce perfect hash families of higher index. Furthermore, the method runs in polynomial time, provided that v, t, λ are fixed.

We briefly mention other computational approaches of PHFs. In astounding work, Moser and Tardös [59] develop a constructive method for objects that matches the bound provided by the LLL. Their method is randomized and runs in polynomial time, if the conditions of the LLL are satisfied. The high-level idea of the algorithm (in the context of hash families) is to select each entry uniformly at random, and if a t-set of columns is not separated, then resample all entries in those t columns; finally, repeat the procedure until all t-sets of columns are separated. However, this method suffers from being (1) randomized, and therefore running in *expected* polynomial time; and (2) the random selections made may cause other t-sets that were previously separated to not be any more.

A more commonly used approach involves constructing the hash family one row at-a-time, while removing t-sets that become separated from a maintained list once a row separates them. This is advantageous in that there is definite progress in constructing the hash family, but disadvantageous in that the $\binom{k}{t}$ sets of columns need to be stored in memory, and may produce more rows than what is guaranteed by Moser and Tardös.

However, we can generate one row at a time in a straightforward way: randomly generate a row until one separates at least the expected number of t-sets to be separated for the first time in this row. Once this happens, append the row to the current array.

This one row at-a-time approach still is randomized, so we outline how to derandomize it, which is given by Colbourn [25]. Define the *density* of a row r to be the expected number of newly separated t-sets in r. A *partial row* is one that contains a symbol $\star$ in addition to the other symbols used in the PHF; such a symbol is to indicate that it is "not yet determined." We start with a partial row consisting entirely of $\star$ entries. If A, B are partial rows, then we indicate $A \to B$ when changing one of the $\star$ entries in A to a non-$\star$ entry in B (with all other entries identical). A *fill sequence* is a sequence of the form $R_k \to R_{k-1} \to \cdots \to R_0$, where R_i is a partial row for all $0 \leq i \leq k$. Since R_k has k $\star$ entries, we must have that R_0 has no $\star$ entries. If we can guarantee that the density of R_{i-1} is at least that of R_i for all i, then we have successfully produced a desired row.

Suppose we are at the ith stage, and have R_i. Pick any $\star$ entry in R_i; there are v ways of assigning a non-$\star$ symbol to this entry. For each of the $\binom{k-1}{t-1}$ ways of selecting $t-1$ other indices, and for each of the v symbols that may be assigned, calculate the expected number of t-sets separated. Now assign the symbol that maximizes this expectation; the resulting row, after fixing all k entries, is R_{i-1}. Since there must be some symbol for which assigning it gives a partial row with density at least the expectation, the density of R_{i-1} is at least that of R_i.

The technique of *column extension* (CE) is, given a hash family, to append as many columns as possible while retaining the separation property. It is effective because assuming that the starting PHF separates all of its t-sets, only the ones involving the new column(s) need to be considered. If one column is added at a time, then this number is $\binom{k}{t-1}$, much smaller than $\binom{k}{t}$. One can then generate a simple randomized procedure for column extension: for a $\mathsf{PHF}(N; k, v, t)$ A, append a new column c to A consisting of entries uniformly sampled from $\{1, \cdots, v\}$. For all $\binom{k}{t}$ t-sets S that involve c, check if some row separates S; if not, then sample c again (or at least until a limit is reached). But if all such sets are separated, then one can repeat the procedure on the resulting $\mathsf{PHF}(N; k+1, v, t)$. If the limit is reached, then a randomly chosen row is added, and repeat until a $\mathsf{PHF}(N'; k+1, v, t)$ with $N' > N$ is formed. Such a process has been used for CPHFs [34] and improvements for small-strength PHFs have been found [45]. What makes CE powerful is not only because the number of columns possible within a reasonable amount of computation time is larger than that of density, but in the many improvements that can be made via recursive techniques, most notably composition (because increasing columns for the first ingredient allows the other ingredient to use more symbols).

The issue with higher-index PHFs is providing a natural generalization of the density metric that meets the probabilistic bounds. A key idea is that the probabilistic bounds guarantee a PHF_λ exists. Suppose R rows have been generated, and the probabilistic method guarantees a PHF_λ on at most $N > R$ rows. Therefore, we generate the PHF_λ, one row at a time as before, but instead generate the "best" symbol at each position *so that the* PHF_λ *can be completed in* $N - R$ *rows provided this symbol is chosen.* The next subsection describes the algorithm more precisely.

3.4.1 Details of the Density Algorithm for Higher-Index

Let A be an array with fewer than N rows (where N is given by the probabilistic method), exactly k columns, over v symbols, and let T be a t-set of columns. Define $\phi(T, A)$ to be the number of times that T is separated in array A, and $p = \binom{v}{t} t!/v^t$.

Suppose that $|A|$ rows have been built, and we are to construct another row ρ; without loss of generality, suppose that ρ is partially built, and we want to assign a value for column c in ρ. We iterate over all t-sets that have not been separated at least λ times; suppose this collection of sets is $\mathcal{T}$, and the observed t-set is T. If T currently has a duplicate in ρ, then it is impossible for T to be separated one more time in ρ, regardless of whether there exist any unfixed entries corresponding to T. Otherwise, there is a probability p that T will be separated in ρ, depending on the number of unfixed entries corresponding to T.

At this point, the method is very similar to the density algorithm of Colbourn. However, we make the following addition; instead of separating at least the average number of t-sets in ρ, we separate as much as possible *so that when (at most) N rows are constructed, strictly less than 1 t-set will remain unseparated* (in expectation if all remaining entries are chosen at random). If T is separated $\phi(T, A)$ times before the addition of ρ, then T's expected number of times remaining to be separated is $\max(0, \lambda - (p \times (\phi(T, A) + 1) + (1 - p) \times \phi(T, A)))$ So, we then calculate the expected number of t-sets not λ-separated within the remaining rows to be constructed. This method is illustrated in Algorithm 1.

For computational efficiency, we set the first row to have value $i \pmod v$ for each column i; in other words, the first row cycles through the symbols so that each symbol appears as equally often as possible. Also, we iterate through the t-sets of columns in *colexicographic order*, which is placing a t-set T_1 before T_2 if and only if the largest element in $T_2 \setminus T_1$ is larger than that of $T_1 \setminus T_2$. This way, for each column index $i \geq t - 1$, each t-set having all of its columns at most i will be examined before any t-set with some index $\geq i + 1$. This method has been used successfully in the context of CPHFs [34].

Lemma 3.6. *In each iteration of the* `while` *loop of Algorithm 1, at least one t-set that is not λ-separated will be separated in the row that is generated.*

Algorithm 1 One Row-At-A-Time Method to Produce PHFs of higher index.

procedure ConditionalExpectation(k, v, t, λ)

  $A \leftarrow$ empty array.

  $p \leftarrow \binom{v}{t} t!/v^t$.

  Set N to be the smallest value so that $\binom{k}{t} \sum_{i=0}^{\lambda-1} \binom{N}{i} p^i (1-p)^{N-i} < 1$.

  Set $g(x, T, N, A) = \sum_{i=0}^{\lambda-\phi(T)-x} \binom{N-|A|-1}{i} p^i (1-p)^{N-|A|-i-1}$.

  while some t-set T is separated fewer than λ times in A **do**

    $\mathcal{T} \leftarrow$ the set of t-sets not λ-separated in A.

    $\rho \leftarrow$ a row of k indeterminates.

    for each column $1 \leq c \leq k$ in any order **do**

      for each value $1 \leq s \leq v$ in any order **do**

        for each $T \in \mathcal{T}$ in any order **do**

          $f \leftarrow$ the number of columns of T fixed in ρ, not including c.

          if ρ has a duplicate in the fixed values corresponding to T including setting s in column c of ρ **then**

            $\chi(T, s) \leftarrow 0$.

          else

            $\chi(T, s) \leftarrow (v - f) \cdots (v - f + 1)/v^{t-f}$.

          end if

        end for

        $calc_s \leftarrow \sum_{T \in \mathcal{T}} \chi(T, s) \cdot g(2, T, N, A) + (1 - \chi(T, s)) \cdot g(1, T, N, A)$.

      end for

      $\rho[c] \leftarrow$ any symbol s with smallest $calc_s$ value.

    end for

    Append ρ to A, and update $\mathcal{T}$ accordingly.

    $N \leftarrow$ smallest value such that $\sum_{T \in \mathcal{T}} g(1, T, N, A) < 1$.

  end while

end procedure

Proof. Suppose a row ρ is generated that does not separate any t-set once. Let A be the array before the addition of ρ, and let A' be derived from appending ρ to A. Since $g(1, T, N, A) < g(1, T, N, A')$, this is a contradiction because the values picked in ρ are such that they do not increase the expected number of unseparated t-sets for the remaining rows, and having 1 fewer row and the same t-sets left would always increase the expectation. $\square$

Theorem 3.15. *Algorithm 1 generates a* $\mathsf{PHF}_\lambda(N; k, v, t)$ *where* N *obeys the bound of Theorem 3.13, and is asymptotically less than that of Theorem 3.14.*

Proof. First note that Algorithm 1 does in fact generate a PHF_λ (because at least one t-set is separated once in each iteration by Lemma 3.6, and there are finitely many of them not λ-separated), so it suffices to prove that the bound for N is met. Furthermore, since N is explicitly set to be the smallest value for which the bound is met after each row is constructed (as well as the smallest value of N that satisfies the bound at the start of the algorithm), it suffices to show that the updated value of N at the end of the `while` loop never increases.

This is true because at least one value s will have its associated $calc_s$ value be at most the expected number of unseparated t-sets if the rest of the to-be-built entries are randomly determined (depending on whether or not the considered t-set is separated one more time in the row being built), so the expected number of unseparated t-sets always is strictly less than 1. Since a PHF_λ is produced, then it must have exactly 0 unseparated t-sets, because the number of such sets always is an integer for an explicit array. Therefore, the number of rows produced is at most the stated bound. $\square$

Theorem 3.16. *Let* v, t, λ *be fixed. Then Algorithm 1 generates a* $\mathsf{PHF}_\lambda(N; k, v, t)$ *in time polynomial in* k.

Proof. We first claim that $g(x, T, N, A)$ can be computed in polynomial time, when x

is fixed, as it is equal to either 1 or 2 in the algorithm. There are a constant number of choices of i, since λ is fixed, and the values $p^i, (1-p)^{N-|A|-i-1}$ can be computed in polynomial time due to repeated squaring. The value of p can be computed in constant time because v, t are fixed, and p does not change value throughout the algorithm. Furthermore, $\binom{N-|A|-i}{i}$ can be computed in polynomial time because there are only a constant number of multiplications being performed, and because N is polynomial in the size of k by either Theorem 3.13 or Theorem 3.14: both bounds are at most $\lambda \log k$, and since λ is fixed, then N is at most $O(\log k)$.

Since t is fixed, there are only a polynomial number of t-sets (in k). Therefore, the polynomial-time guarantee almost follows from the analogous proof by Colbourn [25], with the exception of the last step in Algorithm 1. That calculation can be performed in polynomial time because the sum involves polynomially many terms, and the smallest value of N that satisfies the equation can be found in polynomial time (in the size of k) by binary search, since N will obey the bound of Theorem 3.13. □

An illustration of the bounds of Theorem 3.14 (blue), Theorem 3.13 (red), and that arising from Algorithm 1 (black) are presented in Figure 3.1, provided that λ is sufficiently small relative to k. When k is small, the red line will be smaller than the blue line. However, when k is sufficiently large, the blue line will be smaller than the red line. At all choices of k, the black line will always be smaller than the minimum of the two other lines. In addition, although we do not know the exact asymptotics of PHFN_λ, they must obey a bound that is similar to that of the blue line, since $\log k + \lambda \leq \mathsf{PHFN}_\lambda \leq \log k + \lambda \log\log k + o(\lambda)$, and $\log\log k$ is relatively small compared to k.

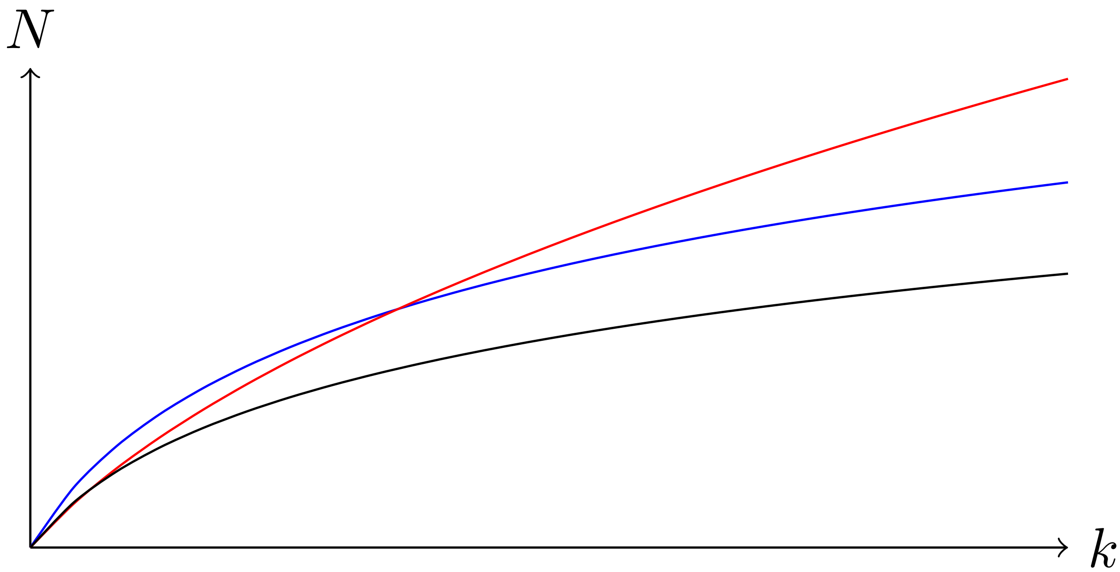

Figure 3.1: Example Asymptotics Rrom Theorem 3.14 (Blue), Theorem 3.13 (Red), and Algorithm 1 (Black), Provided the Index Is Sufficiently Small Relative to the Number of Columns.

3.4.2 Computational Results of the Conditional Expectation Algorithm

Here we showcase results from generating PHF_λs using Algorithm 1. We chose $3 \leq t \leq 6$ because these small strengths are useful in practical domains (see [23]). For each choice of t, we specified a maximum number of columns k_t and different symbol choices for each t. For $t = 3$, we chose $v \in \{3, 4, 5, 6\}$; for $t = 4$, we chose $v \in \{4, 6, 8\}$; for $t = 5$, we chose $v \in \{5, 10\}$; and for $t = 6$, we chose $v \in \{6, 12\}$. The choices of v for strengths 4, 5, and 6 are to showcase a significant difference in the number of rows produced (compared to when $v = t$), but to highlight the "logarithmic curve" in the scatter plots. All different choices of v were not selected here for $t \in \{4, 5, 6\}$ because a small number of additional symbols would not highlight much of a difference in the value PHFN_λ as much as when $t = 3$; however, we expect the curve to have the same shape for non-selected v, along with higher λ choices be somewhat better than scaling the $\lambda = 1$ plot up by the associated index.

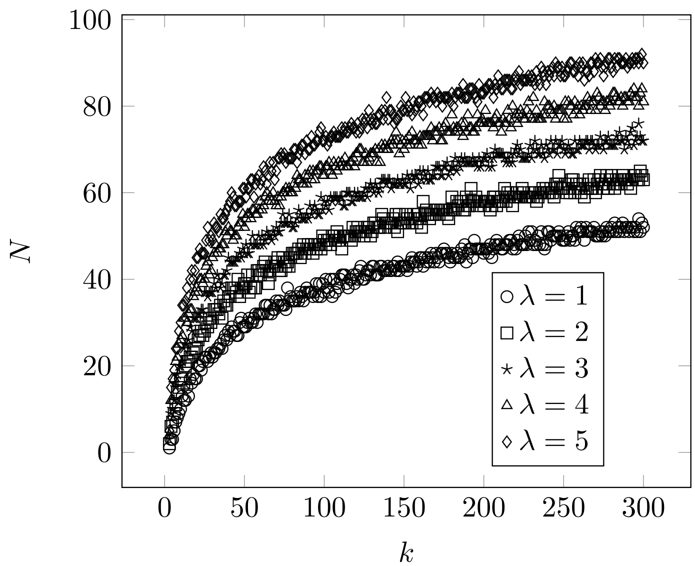

Figure 3.2: Conditional Expectation Results for at Most 300 Columns, 3 Symbols, Strength 3, and Index at Most 5.

We generated PHF_λs for all k and corresponding number of symbols v such that $v \leq k \leq k_t$, $1 \leq \lambda \leq 5$. Our choices of maximum columns were $k_3 = 300, k_4 = 100$, $k_5 = 55$, and $k_6 = 40$. These choices of k_t were chosen as an approximation for $\binom{k_t}{t} = \binom{k_{t+1}}{t+1}$. The results are shown in Figures 3.2 to 3.12.

Recall that Algorithm 1 initially chooses the estimate such that strictly less than 1 t-set will remain not λ-separated. However, the results in Figures 3.2 to 3.12 indicate that this estimate is not often close to the actual number of rows produced. What we can do instead is find a "good" estimate N, regardless of the initial expected number of unseparated t-sets *initially*, such that *at the end, no t-sets remain unseparated*. Essentially, we are "bypassing" the need to update the estimate at each iteration of the `while` loop so that "better" symbols can be chosen in earlier rows.

Furthermore, one would expect that the number of rows would be smaller because for the first few rows, the value of $g(x, T, N, A)$ would make a better judgment of what symbol to place in each entry. Note that Algorithm 1 is a *one row at a time* method, and hence when a row is produced, it is never modified. So if N initially is

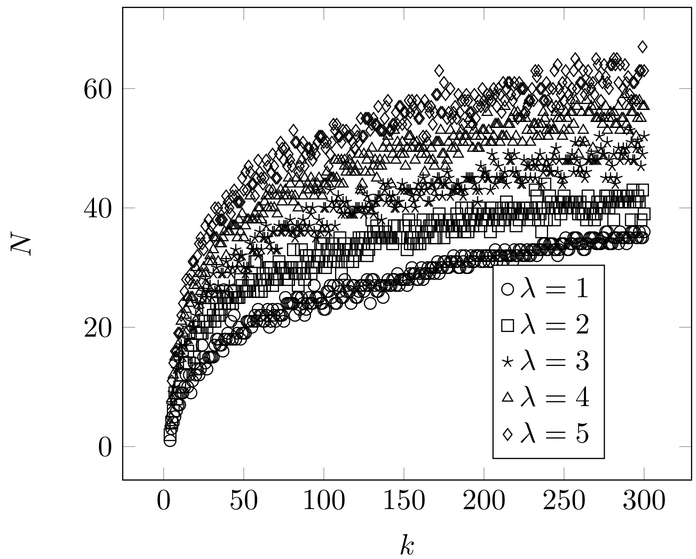

Figure 3.3: Conditional Expectation Results for at Most 300 Columns, 4 Symbols, Strength 3, and Index at Most 5.

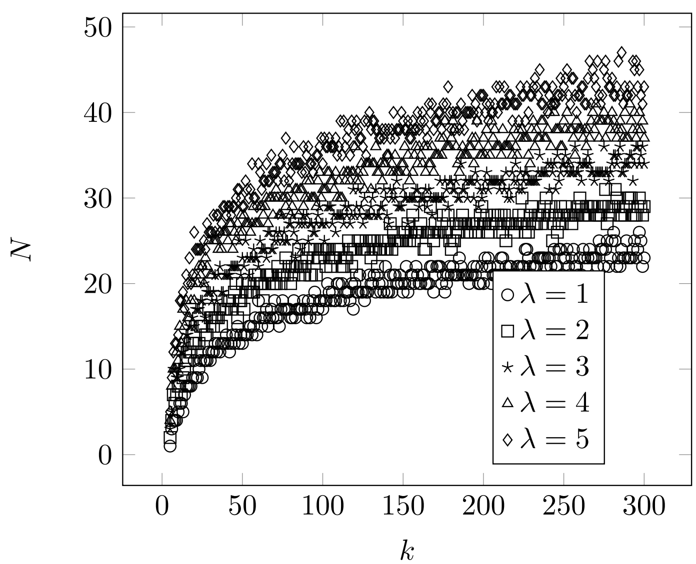

Figure 3.4: Conditional Expectation Results for at Most 300 Columns, 5 Symbols, Strength 3, and Index at Most 5.

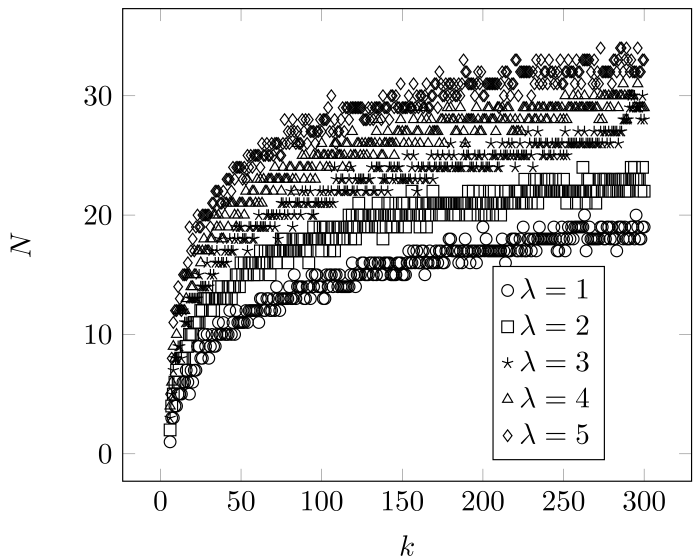

Figure 3.5: Conditional Expectation Results for at Most 300 Columns, 6 Symbols, Strength 3, and Index at Most 5.

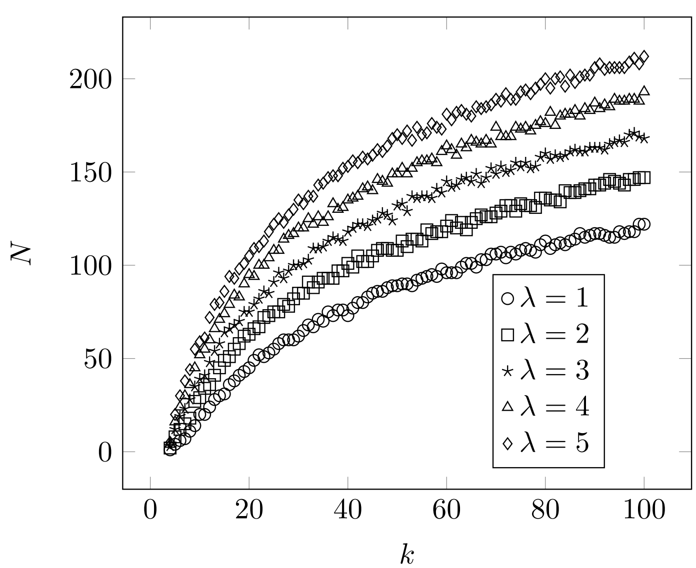

Figure 3.6: Conditional Expectation Results for at Most 100 Columns, 4 Symbols, Strength 4, and Index at Most 5.

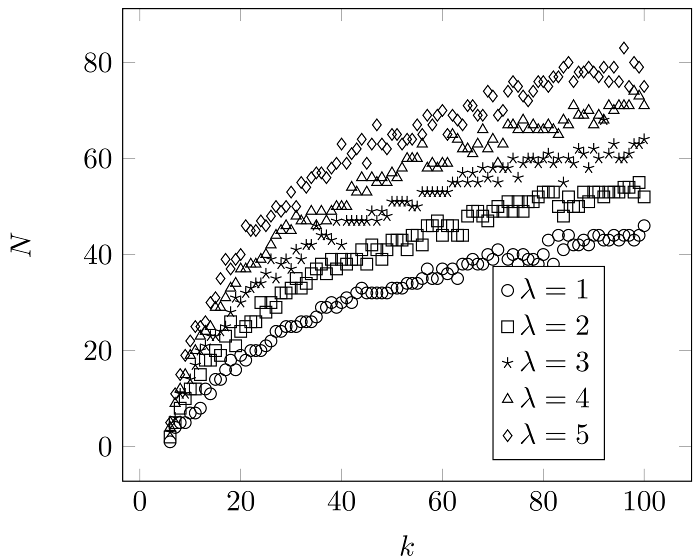

Figure 3.7: Conditional Expectation Results for at Most 100 Columns, 6 Symbols, Strength 4, and Index at Most 5.

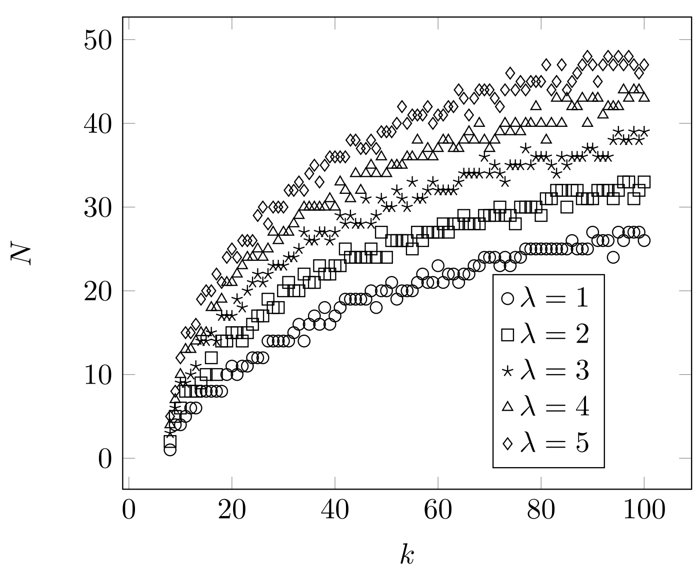

Figure 3.8: Conditional Expectation Results for at Most 100 Columns, 8 Symbols, Strength 4, and Index at Most 5.

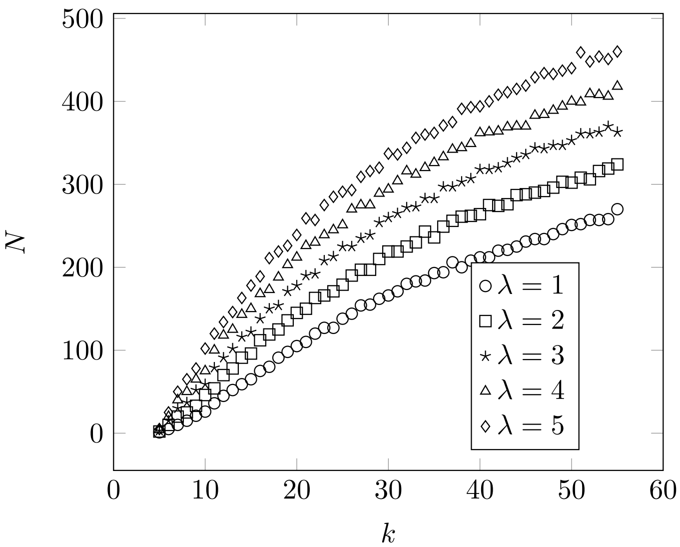

Figure 3.9: Conditional Expectation Results for at Most 55 Columns, 5 Symbols, Strength 5, and Index at Most 5.

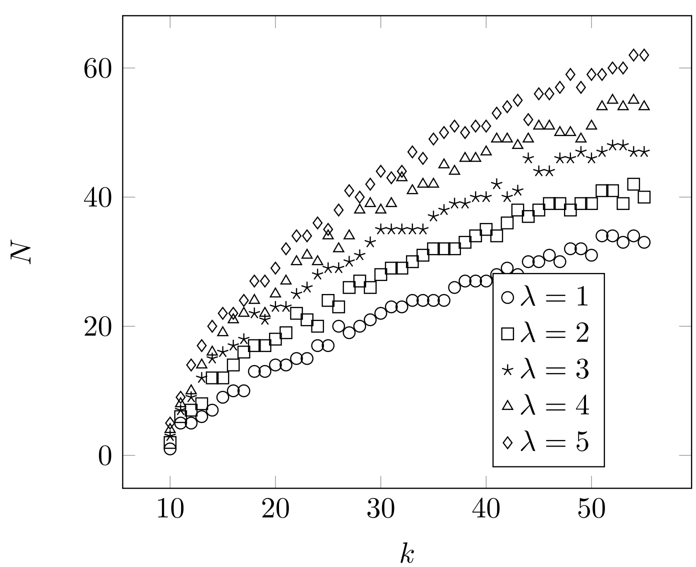

Figure 3.10: Conditional Expectation Results for at Most 55 Columns, 10 Symbols, Strength 5, and Index at Most 5.

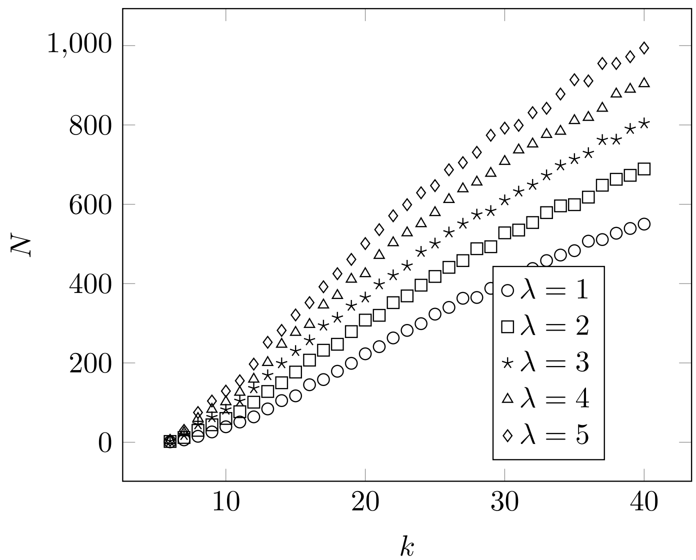

Figure 3.11: Conditional Expectation Results for at Most 40 Columns, 6 Symbols, Strength 6, and Index at Most 5.

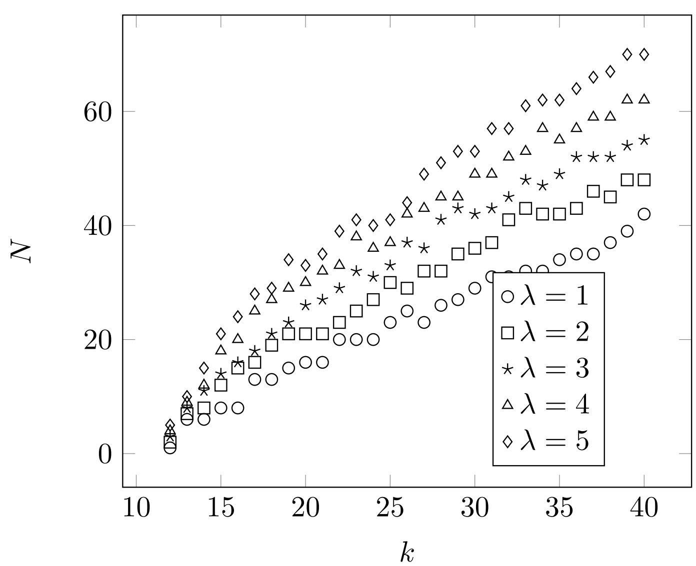

Figure 3.12: Conditional Expectation Results for at Most 40 Columns, 12 Symbols, Strength 6, and Index at Most 5.

an estimate that is very far from the truth, symbol choices in earlier rows may cause more rows at the end to be formed. This modification of Algorithm 1 is given in Algorithm 2.

Algorithm 2 Updated One Row-At-A-Time Method to Produce PHFs of higher index.

1: **procedure** UPDATEDCONDITIONALEXPECTATION(k, v, t, λ)

2: $N \leftarrow 1$.

3: Run Algorithm 1 with parameters k, v, t, λ, and N to be the initial estimate (where N is not modified throughout that algorithm).

4: Repeatedly run the previous step until it successfully generates a PHF_λ with these parameters (by finding the "correct" choice of N via binary search).

5: **end procedure**

However, in all examples of PHFs generated via Algorithm 2, the number of rows produced is nearly identical to that of Algorithm 1. We illustrate this with Tables 3.3 and 3.4, wherein PHF_4s were generated with $k \leq 50$, $v = t = 3$. Both Algorithms 1 and 2 were tested, and the number of rows is produced in these tables. The last column contains the initial expected number of unseparated t-sets for Algorithm 2.

We can see that neither algorithm is the true winner here for all $k \leq 50$, but rather both algorithms are better some of the time. We give a possible explanation for why this occurs. Recall the failure probability for PHF_λs again: $\sum_{i=0}^{\lambda-1} \binom{N}{i} p^i (1-p)^{N-i}$. Even if the number of rows is relatively small compared to the number of rows produced in these tables, then this failure probability is still quite small. This is indicative of how PHFs, at least in the first few rows, have many choices of what symbols to place. After sufficiently many rows are added, and only a few t-sets remain to be separated, then what symbol to place makes much more of a difference. This reasoning is essentially why these two algorithms perform very similarly: after many rows are added, Algorithm 1 has the given estimate much closer to what Algorithm 2's estimate would be after the same number of rows. Why there is a difference in the number of rows entirely is dependent on the estimate of N, and how there the two estimates are not initially similar to each other.

3.5 Conclusion

In this chapter, we investigated perfect hash families of higher index, specifically how to construct them (both algorithmically and with a new recursive construction), bounds on their sizes.

In addition to ensuring that every t-set of columns be separated at least λ times, one might address the more stringent requirement that every t-set be separated at least $\underline{\lambda}$ and at most $\overline{\lambda}$ times. When $\underline{\lambda} = \overline{\lambda}$, such a PHF is *perfectly balanced* [3]. Alon and Gutner [3] establish that a perfectly balanced $\mathsf{PHF}_\lambda(N; k, v, t)$ can exist only when $N = \Omega(k^{\lfloor t/2 \rfloor})$ for t fixed. Contrast this with the $\Theta(\log k)$ growth rate for PHF_1s to understand why perfectly balanced PHFs are not frequently used. On the other hand, a $\mathsf{PHF}(N; k, v, t)$ is *δ-balanced* for some $\delta \geq 1$ if there is a value $T > 0$ so that every t-set of columns is separated at least $\frac{T}{\delta}$ and at most δT times [4]. Alon and Gutner [4] show that for any fixed $\delta > 1$, there is a δ-balanced $\mathsf{PHF}(N; k, v, t)$ with N close to $2^{O(t \log \log t)} \log k$; so, for fixed t, the growth rate is the same as for PHF_1s. Their approach relies (in small part) on the binomial distribution of the number of times a t-set is separated and the application of Chernoff bounds. Moreover, their techniques yield an explicit construction method in principle; its practical effectiveness for intermediate values of k has not been explored.

When δ-balanced PHFs are used in Theorem 2.1 with different t-restrictions, the array constructed inherits from the balanced PHF a lower bound on the number of rows in which the t-restriction is met. However, the t-restriction may be met in a

k	N (Algorithm 1)	N (Algorithm 2)	Initial Unseparated t-Sets (Algorithm 2)
4	12	12	2.93
5	12	16	6.20
6	16	15	9.16
7	21	20	11.21
8	24	21	15.77
9	23	23	13.54
10	27	26	10.47
11	26	27	16.86
12	28	28	16.35
13	30	29	18.06
14	32	32	11.69
15	33	33	10.30
16	36	35	17.99
17	35	36	15.39
18	36	38	15.46
19	38	36	21.93
20	39	38	18.04
21	38	40	14.63
22	40	38	11.70
23	38	39	28.03
24	40	42	12.75
25	38	41	25.30
26	42	44	16.38
27	44	43	18.43

Table 3.3: Comparison of Algorithm 1 and Algorithm 2 With at Most 27 Columns, 3 Symbols, Strength 3, and Index 4.

k	Algorithm 1	Algorithm 2	Estimated Number of Unseparated t-Sets
28	44	43	20.64
29	43	45	15.77
30	44	45	21.19
31	46	47	16.03
32	45	46	21.41
33	47	47	16.06
34	47	46	14.52
35	46	47	19.26
36	50	48	17.32
37	49	50	12.77
38	50	48	24.83
39	50	50	12.34
40	50	49	16.23
41	49	50	21.29
42	51	52	15.50
43	49	52	13.68
44	50	52	12.04
45	53	51	23.31
46	52	52	11.31
47	53	52	17.97
48	54	52	19.17
49	53	52	24.87
50	53	53	21.72

Table 3.4: Comparison of Algorithm 1 and Algorithm 2 With Between 28 and 50 Columns, 3 Symbols, Strength 3, and Index 4.

row arising from a row of the PHF despite failure of the PHF to separate in this row; hence balanced PHFs need not result in balanced t-restrictions through Theorem 2.1. For these reasons, it is reasonable to focus on extending known methods, and finding new methods, for constructing perfect hash families of index $\lambda > 1$. We hope that this chapter would inspire future research in PHFs of higher index.

Chapter 4

FRACTAL HASH FAMILIES

In this chapter, we consider the relationship between the maximum number of columns and the number of symbols for when the number of rows for a hash family is relatively small; precisely, when $N < t$. A theorem of Walker and Colbourn [79] shows that when $k > v$, then $N \geq \lceil \frac{t+1}{2} \rceil$. When $N < t$, as the number of symbols approaches infinity, the ratio $\frac{k}{v}$ approaches a constant, provided that k is as large as possible relative to v. Therefore, in this situation, the maximum number of columns possible is linear in the number of symbols. There are three cases:

- k is superlinear in v when $N \geq t$;
- k is linear in v when $t > N \geq \lceil \frac{t+1}{2} \rceil$; and
- $k = v$ otherwise.

A theorem of Blackburn investigates the asymptotics of the second case, when k is linear in v. The contributions of this chapter are as follows. First, the method is generalized from homogeneous hash families (in which every row has the same number of symbols) to heterogeneous ones. Second, the extension treats distributing hash families, in which only separation into a prescribed number of parts is required, rather than perfect hash families, in which columns must be completely separated. Third, the requirements on one of the main ingredients are relaxed to permit the use of a large class of distributing hash families, which we call *fractal*. Constructions for fractal perfect and distributing hash families are given, and applications to the construction of perfect hash families of large strength are developed.

This chapter has been published in [29], and is currently accepted in [30], for when $\lambda = 1$. We provide a generalization to higher index hash families in Theorem 4.6.

4.1 Linear Bounds on Numbers of Columns

One of the most common uses of perfect hash families is in the construction of covering arrays, as suggested by Theorem 2.1, and some improvements in the number of rows in the formed covering array can be made. Colbourn [26] generalized this theorem further to employ distributing hash families.

Theorem 4.1. *Let $k \geq \min(t, v)$. Suppose that there exist a $\mathsf{DHF}(M; \ell, k, t, \min(t, v))$ and a $\mathsf{CA}(N; t, k, v)$ having ρ constant rows. Then a $\mathsf{CA}(\rho + (N - \rho)M; t, \ell, v)$ exists.*

Colbourn and Torres-Jiménez [38] improved upon Theorem 4.1 in two ways: judiciously choosing symbols on which to place the constant rows (i.e., a row that contains exactly one symbol), and using heterogeneous hash families.

Theorem 4.2. *Suppose that there exist*

1. *a $\mathsf{CA}(N_i; t, k_i, v)$ having ρ_i constant rows and $k_i \geq t$ for $1 \leq i \leq c$, and*

2. *a $\mathsf{DHHF}(M; \ell, k_1^{u_1} \cdots k_c^{u_c}, t, \min(t, v))$.*

Let $\chi = \max(0, v - \sum_{i=1}^{c} u_i(v - \rho_i))$. Then a $\mathsf{CA}(\chi + \sum_{i=1}^{c} u_i(N_i - \rho_i); t, \ell, v)$ exists.

Effective applications of Theorems 2.1, 4.1 and 4.2 require that both the covering arrays and the hash families employed have a "small" number of rows. We investigate families of this form further. A method of Blackburn [15] establishes:

Lemma 4.1. *For positive integers $a_1, \ldots a_t$, set $\tau = \prod_{i=1}^{t} a_i$, and $b_i = \frac{\tau}{a_i}$ for $1 \leq i \leq t$. A $\mathsf{PHHF}(t; \tau, (b_1, \ldots, b_t), t)$ exists in which every set of $1 \leq \ell \leq t$ columns is separated in at least $t + 1 - \ell$ rows.*

Proof. Form a $t \times \tau$ array A, indexing columns by $\{1, \dots a_1\} \times \dots \times \{1, \dots a_t\}$. Form row j by ensuring that two columns contain the same symbol if and only if their indices agree in all coordinates other than the jth coordinate. Suppose to the contrary that for some $1 \leq \ell \leq t$, at most $t - \ell$ rows separate the ℓ columns $c_1, \dots, c_\ell$. Form an edge-coloured graph G on vertex set $\{c_1, \dots, c_\ell\}$; for each row r that does not separate the ℓ columns, place an edge of colour r between two column indices whose columns contain the same symbol in row r. Then G has ℓ vertices and at least ℓ edges each having a different colour. So G contains a cycle $(x_0, \dots, x_{s-1})$ for some $s \leq \ell$. Suppose that edge $\{x_0, x_{s-1}\}$ has colour r. Then the columns indexed by x_0 and x_{s-1} are the same in all rows other than r but differ in row r. On the other hand, for $0 \leq i < s - 1$, edge $\{x_i, x_{i+1}\}$ does not have colour r, and hence the columns agree in row r. This is a contradiction because the columns indexed by x_0 and x_{s-1} must both agree and disagree in row r. □

Lemma 4.1 produces a $\mathsf{PHF}(t; a^t, a^{t-1}, t)$ and hence a $\mathsf{DHF}(t; a^t, a^{t-1}, t, p)$ for every $a \geq 2$ and $t \geq p$. Hence the maximum number of columns grows superlinearly in the number of symbols for DHHFs with $t \geq p \geq 2$ whenever the number of rows is at least t. We are primarily interested in cases where the number of rows is less than the strength t. In these cases, the number of columns cannot exceed a linear function of the number of symbols:

Lemma 4.2. *Let $t \geq p \geq 2$ and $t > n$. If a $\mathsf{DHHF}(n; k, (w_1, \dots, w_n), t, p)$ exists, $k \leq \sum_{i=1}^{n} w_i$.*

Proof. We adapt an argument from [13]. Let $\mathbf{w} = (w_1, \dots, w_n)$. When $p \geq 3$, a $\mathsf{DHHF}(n; k, \mathbf{w}, t, p)$ is also a $\mathsf{DHHF}(n; k, \mathbf{w}, t, p-1)$, so consider a $\mathsf{DHHF}(n; k, \mathbf{w}, t, 2)$, say A. If $n = 1$, any repetition in the single row prevents the array from being a DHHF; hence $k \leq w_1$. Otherwise choose a row having the fewest symbols, say without loss of generality the first row having w_1 symbols. Choose $m \leq w_1$ columns so that in the first row, every symbol that occurs among the columns not chosen also appears among the columns chosen. By deleting the first row, and the m chosen columns, we obtain an $(n-1)\times(k-m)$ array B that we claim is a $\mathsf{DHHF}(n-1; k-m, (w_2, \ldots, w_n), t-1, 2)$. Provided this claim holds, the lemma follows by induction.

Suppose otherwise that there is a partition into 2 parts $\{C_1, C_2\}$ of some $(t-1)$-set of columns of B that is not separated by any row of B. Let σ be a symbol that appears in one of the columns in C_1 in the first row of A, and let c be one of the chosen columns that contains σ in the first row of A. Then no row of A separates the partition $\{C_1, C_2 \cup \{c\}\}$, a contradiction. □

Lemma 4.2 can be often improved upon, by adapting a method of Blackburn [15] for perfect hash families to treat DHHFs when the number of parts is large enough. Let A be a $\mathsf{DHHF}(n; k, (w_1, \ldots, w_n), n+d, p)$ with $d \geq 1$. Call a cell a *singleton* if the symbol it contains does not occur anywhere else in its row. Form an $n \times k$ matrix B, the *singleton array* of A, setting the entry in row r and column c equal to 1 if the cell (r, c) of A is a singleton, and equal to 0 otherwise.

Lemma 4.3. *Let B be the singleton array of a $\mathsf{DHHF}(n; k, (w_1, \ldots, w_n), n+d, p)$, A, with $p \geq d+1 \geq 2$. Then for every d-set of columns, $\{c_1, \ldots, c_d\}$, of B, there is at least one row r of B in which the entry in row r and column c_i equals 1 for all $1 \leq i \leq d$.*

Proof. Suppose to the contrary that A is a $\mathsf{DHHF}(n; k, (w_1, \ldots, w_n), n+d, p)$ with $p \geq d+1$, B is its singleton array, and for columns $C = \{c_1, \ldots, c_d\}$, no row of B has the d entries in these columns all equal to 1. For each row $r = 1, \ldots, n$, there is a column $c_r \in C$ so that the entry of A in row r and column c_r is not a singleton. Then let $d_r \neq c_r$ be a column index so that in row r, the entries of A in columns c_r

and d_r are the same. Now we form a partition of at most $n+d$ columns of A into at most $d+1$ classes that is not separated by any row of A. First, for every $c \in C$ such that $c = c_r$ for some r, form a class containing just the column index c. Next, form a class $\{d_r : 1 \leq r \leq n\} \setminus \{c_r : 1 \leq r \leq n\}$. It follows that c_r and d_r are in different classes for each $1 \leq r \leq n$, so no row accomplishes this separation. Because $|C| = d$, we have chosen a partition of at most $n+d$ columns of A into at most $d+1$ classes, and hence we have the required contradiction. $\square$

If one singleton from each column of a $\mathsf{DHHF}(n; k, (w_1, \ldots, w_n), n+d, p)$ can be identified, then k singletons are identified and the number of identified singletons in row r is at most w_r for $1 \leq r \leq n$. Using this argument it can be seen that Lemma 4.3 improves on Lemma 4.2 when $p \geq d+1 \geq 2$.

For certain parameters, a stronger conclusion can be obtained via the following argument. Form a multigraph G on vertex set $\{c_r, d_r : 1 \leq r \leq n\}$ with edges $\{\{c_r, d_r\} : 1 \leq r \leq n\}$. When G can be properly coloured with γ colours, the array A cannot be a $\mathsf{DHHF}(n; k, v, n+d, \gamma)$. When on the columns $\{c_1, \ldots, c_d\}$ some rows contain multiple entries that are not singletons, we may be able to choose $\{c_r, d_r\}$ for certain values of r in more than one way, and hence choose G so as to reduce the chromatic number of G.

4.2 Fractal Hash Families

Later we describe a construction for DHHFs with a number of rows smaller than the strength, which uses ingredient hash families that are required to satisfy an additional constraint. The hash families to be introduced always have a number of rows equal to the strength t. Lemma 4.1 produces a $\mathsf{PHF}(t; a^t, a^{t-1}, t)$ whenever $a, t \geq 2$, so the number of columns grows faster than linearly in the number of symbols for a $\mathsf{DHF}(t; k, w, t, p)$ with $t \geq p \geq 2$.

However, we can see that the growth of the number of columns is limited:

Theorem 4.3. [62] *If a* $\mathsf{DHF}(t; k, w, t, p)$ *exists then* $k \leq w^2$*. Moreover, if* $t \geq 4$*,* $k \leq w^2 - w$*.*

Indeed the growth rate is less than quadratic asymptotically:

Theorem 4.4. [69] *Let* $t \geq 4$ *and let* $k(w)$ *be the largest integer for which a* $\mathsf{DHHF}(t; k(w), w, t, p)$ *exists. Then* $k(w)$ *is* $o(w^2)$*.*

A $\mathsf{DHHF}(t; k, (v_1, \ldots, v_t), t, p)$ is *fractal* if $t \leq 2$, or if, for each row j, deleting row j yields a fractal $\mathsf{DHHF}(t-1; k, (v_1, \ldots, v_{j-1}, v_{j+1}, \ldots, v_t), t-1, \min(p, t-1))$. Fractal PHHFs are simply fractal DHHFs with $p = t$. A $\mathsf{DHHF}(t; k, (v_1, \ldots, v_n), t, p)$ is *α-fractal* if it is fractal and at least α rows of the DHHF contain all distinct symbols. The following equivalence is straightforward:

Lemma 4.4. *An array is an α-fractal* $\mathsf{DHHF}(t; k, (v_1, \ldots, v_t), t, p)$ *with* $\alpha \geq 1$ *if and only if it has a row* r *containing all distinct symbols and the remaining rows form an* $(\alpha - 1)$*-fractal* $\mathsf{DHHF}(t-1; k, (v_1, \ldots, v_{r-1}, v_{r+1}, \ldots, v_t), t-1, \min(p, t-1))$*.*

One characterization of fractal DHHFs follows:

Lemma 4.5. *A* $\mathsf{DHHF}(t; k, (v_1, \ldots, v_t), t, p)$ *is fractal if and only if every partition of* ℓ *of its columns into* $\min(p, \ell)$ *classes is separated by at least* $t + 1 - \ell$ *rows.*

Proof. Suppose that there is some set S of ℓ columns with $1 \leq \ell \leq t$ and some partition $\{C_1, \ldots, C_{\min(p,\ell)}\}$ of S into $\min(p, \ell)$ classes, so that exactly $\rho \leq t - \ell$ rows separate the classes. The $t - \rho \geq \ell$ remaining rows, say without loss of generality the first $t - \rho$ rows, do not form a $\mathsf{DHHF}(t - \rho; k, (v_1, \ldots, v_{t-\rho}), t - \rho, \min(\ell, p))$ because none of the rows separates classes $C_1, \ldots, C_{\min(p,\ell)}$. So the $\mathsf{DHHF}(t; k, (v_1, \ldots, v_t), t, p)$ cannot be fractal.

In the other direction, if A is not a fractal $\mathsf{DHHF}(t; k, (v_1, \ldots, v_t), t, p)$, then some set of ℓ rows with $2 \le \ell \le t$, say without loss of generality the first ℓ rows, must yield an array B that is not a $\mathsf{DHHF}(\ell; k, (v_1, \ldots, v_\ell), \ell, \min(p, \ell))$. Let $\{C_1, \ldots, C_{\min(p,\ell)}\}$ be a partition that is separated by no row of B. Then $\{C_1, \ldots, C_{\min(p,\ell)}\}$ is separated in A by at most $t - \ell$ rows. $\square$

4.2.1 *Fractal* DHHF*s*

Lemma 4.6. *Whenever $t < \binom{p+1}{2}$, every $\mathsf{DHHF}(t; k, (w_1, \ldots, w_t), t, p)$ is a $\mathsf{PHHF}(t; k, (w_1, \ldots, w_t), t)$.*

Proof. Let A be an $\mathsf{HHF}(t; k, (w_1, \ldots, w_t))$ that is not a $\mathsf{PHHF}(t; k, (w_1, \ldots, w_t), t)$. Choose columns $\{c_1, \ldots, c_t\}$ not separated by any row of A. Form a multigraph G with t vertices, $\{c_1, \ldots, c_t\}$; for each row, choose a pair of columns c_i and c_j having the same symbol in this row and add $\{c_i . c_j\}$ as an edge. Because G has t edges and $t < \binom{p+1}{2}$ by assumption, G has a proper colouring in p colours. (A simple greedy colouring establishes that if $p + 1$ colours were needed, the number of edges must be at least $\sum_{i=1}^{p} i = \binom{p+1}{2}$.) Let $\{C_1, \ldots, C_p\}$ be the colour classes of a proper colouring in p colours. Then the partition $\{C_1, \ldots, C_p\}$ of t columns of A is not separated by any row of A, so A is not a $\mathsf{DHHF}(t; k, (w_1, \ldots, w_t), t, p)$. $\square$

PHF(4;5,4,4)				
1	1	2	3	4
1	2	2	3	4
1	2	3	3	4
1	2	3	4	4

DHF(4;10,4,4,2)									
1	1	1	2	2	2	3	3	3	4
1	2	3	1	2	3	1	2	3	4
1	2	3	2	3	1	3	1	2	4
1	2	3	3	1	2	2	3	1	4

Table 4.1: A PHF(4;5,4,4) and a DHF(4;10,4,4,2).

By restricting the number of parts, fractal DHHFs can exist with more columns than the corresponding fractal PHHFs. An example is given in Table 4.1. According to Niu and Cao [62], every $\mathsf{HF}(4;k,4)$ that accomplishes every separation of four columns into two classes of size two must have $k \leq 10$, and the array shown accomplishes every such separation with $k = 10$. One can verify that the array also separates all partitions of four columns into one class of size three and one of size one, and hence is a DHF(4;10,4,4,2). Lemma 4.6 ensures that every DHF(4;k,4,4,3) must be a PHF(4;k,4,4). A simple counting argument ensures that a PHF(4;k,4,4) has $k \leq 5$. Hence while any PHF(4;k,4,4) or DHF(4;k,4,4,3) has $k \leq 5$, restricting the number of classes in the partition to two doubles the number of columns possible. We return to this in the concluding remarks.

We mention one construction of DHHFs here:

Lemma 4.7. *A fractal* $\mathsf{DHHF}(3; a_1a_2, (a_1, a_1, a_2), 3, 2)$ *exists whenever* $a_1 \geq a_2$ *are positive integers.*

Proof. Index columns by $\{0, \ldots, a_1 - 1\} \times \{0, \ldots, a_2 - 1\}$, In column (a, b), place a in row 1, b in row 3, and $a + b \pmod{a_1}$ in row 2. $\square$

4.2.2 Construction of fractal PHHF*s*

A sufficient condition for a PHHF to be fractal follows.

Lemma 4.8. *If a* $\mathsf{PHHF}(t; k, (v_1, \ldots, v_t), t)$ *has at most one singleton in each row then it is fractal.*

Proof. We prove the result by induction on t. The result is trivial when $t \leq 2$. Let A be a $\mathsf{PHHF}(t; k, (v_1, \ldots, v_t), t)$ with $t \geq 3$ that has at most one singleton in each row. Let B be the array obtained from A by deleting an arbitrary row of A, say the last without loss of generality. It suffices to show that B is a $\mathsf{PHHF}(t-1; k, (v_1, \ldots, v_{t-1}), t-1)$, because then it will follow that B is fractal by our inductive hypothesis.

Suppose otherwise that there is a $(t-1)$-set T of columns of B that is not separated by any row of B. Since $t-1 \geq 2$ and there is at most one singleton in the last row of A, there is a symbol σ that, in the last row of A, appears in some column in T and also in some other column c that may or may not be in T. Then $|T \cup \{c\}| \in \{t-1, t\}$ and no row of A separates the columns in $T \cup \{c\}$, a contradiction. $\square$

Using Lemma 4.8, many PHHFs can be seen to be fractal. For example, Walker and Colbourn [79] use a greedy construction of "triangle-free, 3-regular, resolvable linear spaces (tfrrls)" to produce many $\mathsf{PHF}(3; k, v, 3)$s having no singletons. Fuji-Hara [51] gives an explicit construction of tfrrls using mutually disjoint spreads in a generalized quadrangle, thereby proving that a $\mathsf{PHF}(3; q^2(q+1), q^2, 3)$ exists when $q \geq 3$ is a prime power. Using generalized quadrangles in Hermitian varieties, he also proved that a $\mathsf{PHF}(3; q^5, q^3, 3)$ exists for q a prime power. Lemma 4.1 produces a fractal $\mathsf{PHF}(t; a^t, a^{t-1}, t)$; for $t = 3$, Fuji-Hara's construction has many more columns, suggesting that the easy method of Lemma 4.1 is far from optimal. See also [69] for further improvements when $t = 3$ and when $t = 4$.

Fractal PHHFs can also be constructed recursively. The next result is based on [79, Theorem 4.8].

Theorem 4.5. *Suppose that a* $\mathsf{PHHF}(t; k, (v_1, \ldots, v_t), t)$ *exists with* $k > t \geq 2$*, and that* ℓ *is a positive integer. Then a* $\mathsf{PHHF}(t+1; \ell k, (\ell v_1, \ldots, \ell v_t, k), t+1)$ *exists. If the* PHHF *of strength* t *is fractal, so is the* PHHF *of strength* $t+1$*.*

Proof. Let $\mathsf{A}_0, \ldots, \mathsf{A}_{\ell-1}$ be copies of the (fractal) $\mathsf{PHHF}(t; k, (v_1, \ldots, v_t), t)$ with symbols renamed such that in each row the sets of symbols in A_i and A_j are disjoint when $i \neq j$. Let B be the $(t+1) \times \ell k$ array, with columns indexed by $\{0, \ldots, k-1\} \times \{0,$

$\dots, \ell-1\}$, obtained from $[\mathsf{A}_0 \cdots \mathsf{A}_{\ell-1}]$ by appending a $(t+1)$st row that contains symbol c in column (c, s) for $0 \le c < k$ and $0 \le s \le \ell - 1$.

Let $T = \{(c_i, s_i) : 1 \le i \le t+1\}$ be a set of $t+1$ column indices of B. If the coordinates $\{c_i : 1 \le i \le t+1\}$ are all distinct, then T is separated in row $t+1$. Otherwise $|\{c_i : 1 \le i \le t+1\}| \le t$ and there is a row r of A that separates the set $\{c_i : 1 \le i \le t+1\}$. Because no columns (c, s_i) and (d, s_j) contain the same symbol unless $i = j$, T is separated in row r of B.

Now we show that B is fractal when A is. Suppose that C is obtained by deleting a row of B. If row $t+1$ is deleted, C is a fractal $\mathsf{PHHF}(t; \ell k, (\ell v_1, \dots, \ell v_t), t)$ because A is fractal. If row i with $1 \le i \le t$ is deleted, then C is obtained by applying the construction of this lemma to the fractal $\mathsf{PHHF}(t-1; k, (v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_t), t-1)$ obtained from A by deleting row i. It therefore suffices to prove the statement when $t = 2$ and this is routine to verify. □

4.3 Blackburn's Method, revised

To form a DHHF with n rows and strength $n + d$ with $1 \le d < n$, we generalize a method due to Blackburn [15]. In order to construct PHFs, he used the 'easy' examples of fractal PHFs from Lemma 4.1 without defining and using the fractal property explicitly. In addition to fractal DHHFs, we require a second ingredient, as suggested by Lemma 4.3.

An (n, m, d, λ)-*covering* of *type* $(\rho_0, \dots, \rho_{m-1})$ is a collection of $n + \lambda - 1$ subsets $\{P_0, \dots, P_{n+\lambda-2}\}$ of $\{0, \dots, m-1\}$ satisfying:

1. $|\{P_r : 0 \le r < n, P_r \ni c\}| = \rho_c$ for $0 \le c < m$; and

2. For every $S \subseteq \{0, \dots, m-1\}$ with $|S| = d$, S is a subset of at least λ sets in $\{P_0, ..., P_{n+\lambda-2}\}$.

When $\lambda = 1$, we denote it as an (n, m, d)-covering.

Theorem 4.6. *Suppose that there exist*

- *an* (n, m, d, λ)*-covering* $\mathcal{P} = \{P_0, \ldots, P_{n+\lambda-2}\}$ *of type* $(\rho_0, \ldots, \rho_{m-1})$*, and*
- *for each* $0 \leq c < m$*, a* ρ_c*-fractal* $\mathsf{DHHF}_\lambda(n + \lambda - 1; k_c, (v_{0,c}, \ldots, v_{n-1,c}), n, p)$ *in which, for* $0 \leq r < n$*, row* r *contains all distinct symbols when* $c \in P_r$*.*

Then there exists a $\mathsf{DHHF}_\lambda(n + \lambda - 1; \sum_{c=0}^{m-1} k_c, (w_0, \ldots, w_{n-1}), n + d, p')$ *where*

$$\begin{aligned} w_r &= \textstyle\sum_{c=0}^{m-1} v_{r,c} \text{ for } 0 \leq r < n, \text{ and} \\ p' &= \left\{ \begin{array}{lll} p & \text{if} & p < n - d \\ n + d & \text{if} & p \geq n - d \end{array} \right\}. \end{aligned}$$

Proof. Let A_c be the ρ_c-fractal $\mathsf{DHHF}_\lambda(n+\lambda-1; k_c, (v_{0,c}, \ldots, v_{n-1,c}), n, p)$ for $0 \leq c < m$. Rename the symbols of each of $\{\mathsf{A}_c : 0 \leq c < m\}$ so that in each row the sets of symbols in A_i and A_j are disjoint when $i \neq j$. Set $\mathsf{B} = [\mathsf{A}_0 \cdots \mathsf{A}_{m-1}]$. Then B has n rows and $\sum_{c=0}^{m-1} k_c$ columns, and for each $0 \leq r < n$, there are w_r different symbols in row r. To show that B is a $\mathsf{DHHF}_\lambda(n + \lambda - 1; \sum_{c=0}^{m-1} k_c, (w_0, \ldots, w_{n-1}), n + d, p)$, it suffices to show that every partition of $(n + d)$ columns into p classes is separated at least λ times.

Consider a $(n + d)$-set T of column indices and a partition $\mathcal{T}$ of T into p' classes. For $0 \leq c < m$, let ℓ_c be the number of columns of A_c in T, and let $\mathcal{T}_c$ be the restriction of $\mathcal{T}$ to the columns of A_c. Because every two of the $\{\mathsf{A}_c : 0 \leq c < m\}$ share no symbols, it suffices to show that there are λ rows of B that separates each partition $\mathcal{T}_c$ for $0 \leq c < m$. Let $L = \{c : \ell_c \geq 2\}$, and let $\nu = |L|$. Note that A_c trivially separates $\mathcal{T}_c$ at least λ times for each $c \in \{0, \ldots, m\} \setminus L$.

If $\nu \leq d$, the (n, m, d, λ)-covering contains λ sets $P_{r_1}, \cdots, P_{r_\lambda}$ with $L \subseteq P_{r_i}$ for $1 \leq i \leq \lambda$. Then in rows $r_1, \cdots, r_\lambda$, for each $c \in L$, A_c contains all distinct symbols and therefore separates the partition $\mathcal{T}_c$.

So suppose that $\nu > d$. Then, for each $0 \le c < m$, $\ell_c \le n - d$ and hence $\mathcal{T}_c$ has at most $\min(p', n-d) \le p$ nonempty classes. By Lemma 4.5, for each $c \in L$, $\mathcal{T}_c$ can fail to be separated in at most $\ell_c - \lambda$ rows of A_c because A_c is a fractal DHHF for p parts. Because $\nu > d$, $\sum_{h=0}^{m-1} \max(0, \ell_h - \lambda) \le (n + \lambda - 1) + d - (\lambda - 1) - \nu = n + d - \nu < n$, and so at least λ rows of B separate each partition $\mathcal{T}_c$ for $0 \le c < m$. $\square$

We employ an easy variant of Theorem 4.6 repeatedly when $\lambda = 1$:

Lemma 4.9. *Suppose that $d \ge 1$ and a $\mathsf{PHHF}(n; \kappa, (w_1, \ldots, w_n), n + d)$ exists.*

(i) *Whenever $\alpha, k \ge 1$, a $\mathsf{PHHF}(n + \alpha; \kappa + \alpha k, (w_1 + \alpha k)^1 \cdots (w_n + \alpha k)^1 (\kappa + (\alpha - 1)k + 1)^\alpha, n + d + 2\alpha)$ exists.*

(ii) *In particular, whenever a $\mathsf{PHF}(n; \kappa, w, n+d)$ exists, a $\mathsf{PHF}(n+\alpha; \kappa+\alpha(\kappa-w+1), \kappa + (\alpha - 1)(\kappa - w + 1) + 1, n + d + 2\alpha)$ exists.*

Proof. Statement (ii) follows from (i) by setting $w_1 = \cdots = w_n = w$ and $k = \kappa+1-w$, so it suffices to prove (i). Furthermore, we only need to deal with the case where $\alpha = 1$ because the remainder of the result follows by induction.

Append a row with κ distinct symbols to the $\mathsf{PHHF}(n; \kappa, (w_1, \ldots, w_n), n + d)$ to form A_0. Form an $(n + 1) \times k$ array A_1 in which row $n + 1$ contains k occurrences of a single symbol, all other rows contain distinct symbols, and the sets of symbols in A_0 and A_1 are disjoint. We claim that $\mathsf{B} = [\mathsf{A}_0 \mathsf{A}_1]$ is the required PHHF.

Consider a $(n + d + 2)$-set T of column indices. If T contains at most one column of A_1, then T is separated by the last row of B. Otherwise, the restriction of T to the columns of A_0 contains at most $n + d$ columns and so is separated by some row r of A_0 other than the last. Then row r of B separates T. $\square$

Between them, Theorem 4.6 and Lemma 4.9 provide a flexible framework for constructing DHHFs. In Lemmas 4.10–4.17 we give more concrete applications of these two results to producing PHFs and PHHFs with strength larger than their number of rows. We conclude the section by considering the asymptotic ratio of columns to symbols in large PHFs constructed by these lemmas.

We begin by choosing the covering in Theorem 4.6 to consist of all d-subsets of an m-set, an $(\binom{m}{d}, m, d)$-covering.

Lemma 4.10. *Let $m > d \geq 1$ be integers. Suppose that a fractal*

$$\mathsf{PHHF}(\tbinom{m-1}{d}; \kappa, (w_0, \ldots, w_{\binom{m-1}{d}-1}), \tbinom{m-1}{d})$$

exists. Let σ be the sum of the $m-d$ largest elements in $\{w_i : 0 \leq i \leq \binom{m-1}{d} - 1\}$. Then a $\mathsf{PHF}(\binom{m}{d}; m\kappa, d\kappa + \sigma, \binom{m}{d} + d)$ exists.

Proof. Let A be the $\mathsf{PHHF}(\binom{m-1}{d}; \kappa, (w_0, \ldots, w_{\binom{m-1}{d}-1}), \binom{m-1}{d})$. Take the $(\binom{m}{d}, m, d)$-covering $\{P_0, \ldots, P_{\binom{m}{d}-1}\}$ in which the sets are all of the d-sets of $\{0, \ldots, m-1\}$. This covering has $\rho_c = \binom{m-1}{d-1}$ for all $0 \leq c < m$. Form a bipartite graph G with vertex set $\{x_0, \ldots, x_{\binom{m}{d}-1}\} \cup \{y_0, \ldots, y_{m-1}\}$, placing an edge between x_r and y_c when the rth d-set does not contain the element c. Note $\deg_G(x_i) = m - d$ for $0 \leq i < \binom{m}{d}$ and $\deg_G(y_i) = \binom{m-1}{d}$ for $0 \leq i < m$. So we can properly edge colour G with $\binom{m-1}{d}$ colours $\{0, \ldots, \binom{m-1}{d} - 1\}$. Now apply Theorem 4.6. For each $0 \leq c < m$, use as an ingredient the $\binom{m-1}{d-1}$-fractal PHHF A_c obtained from A by adding $\binom{m-1}{d-1}$ rows of distinct symbols and rearranging the rows in such a way that, when edge $\{x_r, y_c\}$ of G has colour ℓ, row ℓ of A (with w_ℓ symbols) is row r of A_c. For $0 \leq r < \binom{m}{d}$, row r of the resulting PHF has at most $d\kappa + \sigma$ symbols because $m - d$ distinct colours occur at the vertex x_r of G. $\square$

$$
\begin{array}{cccc|cccc|cccc}
0 & 0 & 1 & 1 & 2 & 2 & 3 & 3 & i & j & k & \ell \\
0 & 1 & 0 & 1 & e & f & g & h & 4 & 5 & 4 & 5 \\
a & b & c & d & 2 & 3 & 2 & 3 & 4 & 4 & 5 & 5
\end{array}
$$

Figure 4.1: A $\mathsf{PHF}(3; 12, 8, 4)$

$$
\begin{array}{cccccccccccc}
0 & 1 & 2 & 1 & 0 & 5 & 6 & 3 & 1 & 7 & 0 & 4 \\
0 & 1 & 2 & 3 & 4 & 5 & 2 & 2 & 6 & 2 & 7 & 5 \\
0 & 1 & 2 & 0 & 1 & 4 & 4 & 5 & 3 & 6 & 3 & 2
\end{array}
$$

Figure 4.2: A $\mathsf{PHHF}(3; 12, (8, 8, 7), 4)$.

To illustrate Lemma 4.10, we provide an example. Here is a $\mathsf{PHF}(2; 4, 2, 2)$:

$$
\begin{array}{cccc}
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1
\end{array}
$$

Note that since there are 2 rows, we have $m = 3, d = 1$. Since the PHF here is homogeneous, then $\sigma = 4$. So we will be forming a $\mathsf{PHF}(3; 12, 8, 4)$. The corresponding $3, 3, 1$-covering is $\{\{0\}, \{1\}, \{2\}\}$, with $\rho_c = 1$ for all c. The bipartite graph formed has 6 vertices, with 3 vertices in each part, and precisely 6 edges: $\{x_0, y_1\}, \{x_0, y_2\}$, $\{x_1, y_0\}, \{x_1, y_2\}, \{x_2, y_0\}, \{x_2, y_1\}$. This graph is isomorphic to C_6, which can be edge-colored with 2 colors, as expected, by alternating between two colors. We now create the three A_c ingredients, obtained by adding 1 row of distinct symbols and observing the edge color formed on the graph, shown in Figure 4.1 (a completely different set of symbols is used for when the rows of distinct symbols are employed in each ingredient). Vertical bars show partition the 12 columns into 3 sets of 4 columns each.

Note that Lemma 4.10 is not optimal even in this case, because there exists a $\mathsf{PHHF}(3; 12, (8, 8, 7), 4)$, shown in Figure 4.2, found by the satisfiability formula described in Section 3.2.3. Two applications of Lemma 4.10, with $d = 1$ and $d = n - 1$, are of particular interest.

Lemma 4.11. *When a fractal* $\mathsf{PHHF}(n-1;\kappa,(w_1,\ldots,w_{n-1}),n-1)$ *exists, a* $\mathsf{PHF}(n;n\kappa,\kappa+\sum_{i=1}^{n-1}w_i,n+1)$ *exists.*

Proof. Apply Lemma 4.10 with $(m,d)=(n,1)$. □

Lemma 4.12. *For all* $n\geq 2$ *and* $\kappa\geq 1$, *a* $\mathsf{PHF}(n;n\kappa,(n-1)\kappa+1,2n-1)$ *exists.*

Proof. Apply Lemma 4.10 with $(m,d)=(n,n-1)$. (The PHHF ingredient has one row and strength 1.) □

Next we give other applications of Theorem 4.6 and Lemma 4.9 to handle cases with $d\in\{n-2,n-3,n-4,n-5\}$.

Lemma 4.13. *Suppose that a* $\mathsf{PHHF}(2;\kappa,(w_1,w_2),2)$ *exists and* $n\geq 3$. *Then*

(i) *When* $k\geq 1$, *a* $\mathsf{PHHF}(n;3\kappa+(n-3)k,(\kappa+(n-3)k+w_1+w_2)^3(3\kappa+(n-4)k+1)^{n-3},2n-2)$ *exists.*

(ii) *When* $w_1+w_2\leq 2\kappa$, *a* $\mathsf{PHF}(n;(2n-3)\kappa-(n-3)(w_1+w_2-1),(2n-5)\kappa-(n-4)(w_1+w_2-1)+1,2n-2)$ *exists.*

Proof. It suffices to prove (i) because (ii) follows from (i) by setting $k=2\kappa+1-w_1-w_2$. Lemma 4.11 establishes (i) when $n=3$. Apply Lemma 4.9(i) with $\alpha=n-3$. □

Lemma 4.14. *Suppose that a* $\mathsf{PHHF}(2;\kappa,(w_1,w_2),2)$ *exists and* $n\geq 6$. *Then*

(i) *When* $k\geq 1$, *a* $\mathsf{PHHF}(n;6\kappa+(n-6)k,(4\kappa+(n-6)k+w_1+w_2)^6(6\kappa+(n-7)k+1)^{n-6},2n-3)$ *exists.*

(ii) *When* $w_1+w_2\leq 2\kappa$, *a* $\mathsf{PHF}(n;(2n-6)\kappa-(n-6)(w_1+w_2-1),(2n-8)\kappa-(n-7)(w_1+w_2-1)+1,2n-3)$ *exists.*

Proof. It suffices to prove (i) because (ii) follows from (i) by setting $k = 2\kappa + 1 - w_1 - w_2$. By Lemma 4.9(i) with $\alpha = n - 6$, it suffices to treat the case when $n = 6$. Form the 4-fractal PHHFs using the numbers of symbols in the columns given:

$$\begin{array}{|c|c|c|c|c|c|}
w_1 & w_2 & \kappa & \kappa & \kappa & \kappa \\
\kappa & w_1 & w_2 & \kappa & \kappa & \kappa \\
w_2 & \kappa & w_1 & \kappa & \kappa & \kappa \\
\kappa & \kappa & \kappa & w_1 & w_2 & \kappa \\
\kappa & \kappa & \kappa & \kappa & w_1 & w_2 \\
\kappa & \kappa & \kappa & w_2 & \kappa & w_1
\end{array}$$

Let $P_0, \ldots, P_5$ be the indices of the κ entries in the six rows. This yields the $(6, 6, 3)$-covering. Apply Theorem 4.6. $\square$

Lemma 4.15. *Suppose that a fractal* $\mathsf{PHHF}(3; \kappa, (w_1, w_2, w_3), 3)$ *exists and* $n \geq 6$. *Then*

(i) *When* $k \geq 1$, *a* $\mathsf{PHHF}(n; 6\kappa + (n-6)k, (3\kappa + (n-6)k + w_1 + w_2 + w_3)^6 (6\kappa + (n-7)k + 1)^{n-6}, 2n - 4)$ *exists.*

(ii) *When* $w_1 + w_2 + w_3 \leq \kappa$, *a* $\mathsf{PHF}(n; (3n-12)\kappa - (n-6)(w_1 + w_2 + w_3 - 1), (3n-15)\kappa - (n-7)(w_1 + w_2 + w_3 - 1) + 1, 2n - 4)$ *exists.*

Proof. It suffices to prove (i) because (ii) follows from (i) by setting $k = 3\kappa + 1 - w_1 - w_2 - w_3$. By Lemma 4.9(i) with $\alpha = n - 6$ it suffices to treat the case when $n = 6$. We use a $(6, 6, 2)$-covering. For $0 \leq j < 6$, let $P_j = \{j, j+1 \bmod 6, j+3 \bmod 6\}$. To form the 3-fractal PHHFs $\{A_0, \ldots, A_5\}$, set

$$v_{rc} = \begin{cases} w_1 & \text{if} \quad 0 \leq r < 6, 0 \leq c < 6, c \equiv r + 2 \pmod 6 \\ w_2 & \text{if} \quad 0 \leq r < 6, 0 \leq c < 6, c \equiv r + 4 \pmod 6 \\ w_3 & \text{if} \quad 0 \leq r < 6, 0 \leq c < 6, c \equiv r + 5 \pmod 6 \\ \kappa & \text{if} \quad 0 \leq c < 6, \text{ and } c \equiv r, r+1, r+3 \pmod 6 \end{cases}$$

Apply Theorem 4.6. $\square$

Lemma 4.16. *Suppose that a fractal* $\mathsf{PHHF}(4;\kappa,(w_1,w_2,w_3,w_4),4)$ *exists and* $n \geq 7$. *Then*

(i) *When* $k \geq 1$, *a* $\mathsf{PHHF}(n;7\kappa+(n-7)k,(3\kappa+(n-7)k+w_1+w_2+w_3+w_4)^7(7\kappa+(n-8)k+1)^{n-7},2n-5)$ *exists.*

(ii) *When* $w_1+w_2+w_3+w_4 \leq 4\kappa$, *a* $\mathsf{PHF}(n;(4n-21)\kappa-(n-7)(w_1+w_2+w_3+w_4-1),(4n-25)\kappa-(n-8)(w_1+w_2+w_3+w_4-1)+1,2n-5)$ *exists.*

Proof. It suffices to prove (i) because (ii) follows from (i) by setting $k = 4\kappa+1-w_1-w_2-w_3-w_4$. By Lemma 4.9(i) with $\alpha = n-7$ it suffices to treat the case when $n = 7$. We use a $(7,7,2)$-covering. When $n = 7$, for $0 \leq j < 7$, let $P_j = \{j, j+1 \bmod 7, j+3 \bmod 7\}$. To form the 3-fractal PHHFs $\{A_0,\ldots,A_6\}$, set

$$v_{rc} = \begin{cases} w_1 & \text{if} \quad 0 \leq r < 7, 0 \leq c < 7, c \equiv r+2 \pmod 7 \\ w_2 & \text{if} \quad 0 \leq r < 7, 0 \leq c < 7, c \equiv r+4 \pmod 7 \\ w_3 & \text{if} \quad 0 \leq r < 7, 0 \leq c < 7, c \equiv r+5 \pmod 7 \\ w_4 & \text{if} \quad 0 \leq r < 7, 0 \leq c < 7, c \equiv r+6 \pmod 7 \\ \kappa & \text{if} \quad 0 \leq c < 7, \text{ and } c \equiv r, r+1, r+3 \pmod 7 \end{cases}$$

Apply Theorem 4.6. $\square$

Finally we treat a special case with $d = 2$.

Lemma 4.17. *If there exist*

- *a* $\mathsf{PHHF}(2;\kappa_2,(v_{1,2},v_{2,2}),2)$,
- *a* $\mathsf{PHHF}(2;\kappa_3,(v_{1,3},v_{2,3}),2)$, *and*
- *a fractal* $\mathsf{PHHF}(3;\kappa_1,(v_{1,1},v_{2,1},v_{3,1}),3)$,

then a $\mathsf{PHHF}(5; 2\kappa_1 + 2\kappa_2 + \kappa_3, (w_0, \ldots, w_4), 7)$ *exists with* $w_0 = \kappa_1 + 2\kappa_2 + v_{1,1} + v_{1,3}$, $w_1 = \kappa_1 + 2\kappa_2 + v_{1,1} + v_{2,3}$, $w_2 = 2\kappa_1 + \kappa_3 + v_{1,2} + v_{2,2}$, $w_3 = \kappa_2 + \kappa_3 + v_{2,1} + v_{3,1} + v_{2,2}$, $w_4 = \kappa_2 + \kappa_3 + v_{1,2} + v_{2,1} + v_{3,1}$.

Proof. Using a fractal $\mathsf{PHHF}(3; \kappa_1, (v_{1,1}, v_{2,1}, v_{3,1}), 3)$, a $\mathsf{PHHF}(2; \kappa_2, (v_{1,2}, v_{2,2}), 2)$, and a $\mathsf{PHHF}(2; \kappa_3, (v_{1,3}, v_{2,3}), 2)$, form five PHHFs on 5 rows by placing the rows as indicated in each column shown; when κ_i is specified, the row is all distinct symbols.

$v_{1,1}$	κ_1	κ_2	κ_2	$v_{1,3}$
κ_1	$v_{1,1}$	κ_2	κ_2	$v_{2,3}$
κ_1	κ_1	$v_{1,2}$	$v_{2,2}$	κ_3
$v_{2,1}$	$v_{3,1}$	$v_{2,2}$	κ_2	κ_3
$v_{3,1}$	$v_{2,1}$	κ_2	$v_{1,2}$	κ_3

Let $P_0, \ldots, P_4$ be the indices of the κ entries in the five rows. This yields the $(5, 5, 2)$-covering. Apply Theorem 4.6. □

Numerous cases have been handled by Lemma 4.10. We could take $m = 5$ and $d = 2$ to yield PHFs with 10 rows and strength 12, or $m = 5$ and $d = 3$ to yield PHFs with 10 rows and strength 13. However, Lemma 4.10 need not yield the best result asymptotically, as shown by the $(10, 10, 2)$-covering with blocks 0169, 2379, 4589, 0178, 2368, 4567, 024, 035, 125, 134. Using ingredients with κ columns on elements $\{0, \ldots, 5\}$, and $\kappa/2$ on elements $\{6, \ldots, 9\}$, the number of columns grows like 8κ while the number of symbols grows like 3κ. Table 4.2 summarizes the best asymptotic ratio of columns to symbols in large PHFs constructed using the lemmas in this section; this extends somewhat a table from [15].

$n \downarrow d \rightarrow$	1	2	3	4	5	6	7	8
2	4.11: 2							
3	4.11: 3	4.12: $\frac{3}{2}$						
4	4.11: 4	4.13: $\frac{5}{3}$	4.12: $\frac{4}{3}$					
5	4.11: 5	4.17: $\frac{9}{5}$	4.13: $\frac{7}{5}$	4.12: $\frac{5}{4}$				
6	4.11: 6	4.15: 2	4.14: $\frac{3}{2}$	4.13: $\frac{9}{7}$	4.12: $\frac{6}{5}$			
7	4.11: 7	4.16: $\frac{7}{3}$	4.15: $\frac{3}{2}$	4.14: $\frac{4}{3}$	4.13: $\frac{11}{9}$	4.12: $\frac{7}{6}$		
8	4.11: 8		4.16: $\frac{11}{7}$	4.15: $\frac{4}{3}$	4.14: $\frac{5}{4}$	4.13: $\frac{13}{11}$	4.12: $\frac{8}{7}$	
9	4.11: 9			4.16: $\frac{15}{11}$	4.15: $\frac{5}{4}$	4.14: $\frac{6}{5}$	4.13: $\frac{15}{13}$	4.12: $\frac{9}{8}$
10	4.11: 10				4.16: $\frac{19}{15}$	4.15: $\frac{6}{5}$	4.14: $\frac{7}{6}$	4.13: $\frac{17}{15}$

Table 4.2: PHFs with Few Rows from Lemmas 4.10–4.17. For Each Case, the Number of the Relevant Lemma, and the Asymptotic Ratio of the Number of Columns to the Number of Symbols Achieved, Is Given.

4.5 Existence Tables

In order to assess the impact of using Blackburn's construction for perfect hash families using fractal ingredients, we have created tables on the best-known upper bounds on $\mathsf{PHFN}(k, v, t)$ for $k \leq 10^9, v \leq 2500$, and $3 \leq t \leq 11$ [45]. These tables report on over 385,000 parameter situations. Of those, 2,658 are improvements that result from the generalization of Blackburn's theorem. Improvements were found only for larger strengths, in particular when $t \geq 6$. We provide here a representative collection of improvements, restricting our attention to cases with $N < t$ and $v < 250$. Each table considers a selection of N and t; then k_{old} is the largest number of columns found without using the fractal version of the Blackburn construction, while $k_{fractal}$ gives the largest number of columns obtained using in addition fractal PHHFs in the Blackburn construction. In order to highlight the more significant improvements, we only report cases when $k_{fractal} \geq k_{old} + 5$. However, we do list all other improvements (i.e., $250 \leq v \leq 2500$, or $k_{fractal} - k_{old} \leq 4$ for all v) in Appendix A. Naturally, other recursive constructions can and do make further improvements, but we do not address them here.

Table 4.3: Improvements for Strength 6, Four Rows

v	121	127	163	166	169	211	217
$k_{fractal}$	188	198	256	260	266	334	344
k_{old}	183	192	247	253	259	326	337

Table 4.4: Improvements for Strength 6, Five Rows

v	50	63	68	75	83	93	101	108	115	121	130	135	140	148
$k_{fractal}$	90	135	140	155	175	205	225	240	255	265	290	295	300	320
k_{old}	82	104	113	125	139	172	197	216	223	229	254	259	264	272
v	157	165	172	181	189	196	207	215	223	228	238	246		
$k_{fractal}$	345	365	380	405	425	440	475	495	515	520	550	570		
k_{old}	281	292	303	313	405	412	423	431	439	444	454	481		

The use of fractal DHHFs and PHHFs in the Blackburn method leads to many constructions for DHHFs with a number of rows less than the strength. Our motivation for seeking these improvements has been to improve bounds for covering arrays. Many improvements are reported in the online covering array tables [24]. We make no effort to enumerate them here, contenting ourselves to mention a few illustrative examples.

Renaming symbols in each column of a covering array, we can always produce at least one constant row. Then by Theorem 4.2, the existence of a $\mathsf{PHF}(4; 260, 166, 6)$ and a $\mathsf{CA}(N; 6, 166, v)$ ensures that a $\mathsf{CA}(4N-4; 6, 260, v)$ exists. This yields the smallest covering array for these parameters when $v \in \{7, 8, 9, 11, 12, 13\}$. Because a $\mathsf{PHHF}(2; ab, (a, b), 2)$ exists whenever $a, b \geq 1$, a $\mathsf{PHHF}(2; 55, (7, 8), 2)$ exists. Then using Lemma 4.13(i) with $k = 95$, there is a $\mathsf{PHHF}(4; 260, 165^3 166^1, 6)$. Hence by

Table 4.5: Improvements for Strength 7, Five Rows

v	78	80	81	82	114	115	123	124	127	128	129	130	131	133
$k_{fractal}$	116	120	121	122	174	175	189	190	194	196	198	201	202	204
k_{old}	111	114	115	117	169	170	183	185	189	190	191	192	193	195
v	134	135	136	137	138	139	141	142	143	144	146	148	149	150
$k_{fractal}$	205	207	208	209	210	214	217	218	219	224	226	228	233	234
k_{old}	196	197	198	199	200	202	204	207	211	216	211	213	217	222
v	154	155	156	157	158	159	161	162	164	167	168	169	170	171
$k_{fractal}$	238	239	240	242	244	246	248	252	257	260	264	265	266	268
k_{old}	226	227	228	229	230	231	237	242	248	251	252	253	254	255
v	172	175	177	179	182	183	184	185	186	187	188	189	190	191
$k_{fractal}$	269	273	276	281	284	287	288	289	290	292	294	297	298	299
k_{old}	256	259	261	266	269	270	272	274	275	277	279	281	283	285
v	192	193	194	195	196	197	201	203	204	206	208	209	210	211
$k_{fractal}$	304	305	306	307	308	312	316	318	320	326	328	329	333	334
k_{old}	287	289	290	281	292	293	308	313	314	316	318	318	320	321
v	212	213	214	216	217	219	221	222	223	224	226	227	229	230
$k_{fractal}$	335	336	340	343	344	346	348	349	350	354	360	362	365	366
k_{old}	322	323	324	326	327	329	331	333	335	337	341	343	347	349
v	231	232	233	234	235	236	242	243	244	245	246	247	248	249
$k_{fractal}$	367	368	369	273	377	380	386	387	388	389	389	391	393	398
k_{old}	351	353	355	357	359	369	372	375	376	378	380	382	384	386

Table 4.6: Improvements for Strength 8, Six Rows

v	108	120	135	158	174	184	195	207	218	227	240
$k_{fractal}$	162	192	216	240	270	288	300	324	336	360	384
k_{old}	156	175	195	233	258	271	291	308	330	339	360

Table 4.7: Improvements for Strength 9, Six Rows

v	173	181	191	194	231	236	239
$k_{fractal}$	240	252	264	270	324	330	336
k_{old}	234	244	256	259	315	320	323

Table 4.8: Improvements for Strength 10, Seven Rows

v	191	215	239
$k_{fractal}$	215	298	323
k_{old}	253	280	318

Table 4.9: Improvements for Strength 11, Seven Rows

v	143	179	191	209	239
$k_{fractal}$	183	230	245	269	308
k_{old}	178	224	239	264	300

Theorem 4.2. if a $\mathsf{CA}(N; 6, 165, v)$ and a $\mathsf{CA}(N'; 6, 166, v)$ both exist, a $\mathsf{CA}(3N + N' - 4; 6, 260, v)$ exists. This illustrates how the use of heterogeneous hash families can reduce the number of rows in the covering array produced.

Using Lemma 4.11 with a fractal $\mathsf{PHF}(4; 81, 25, 4)$ (found by the method of [25]) yields a $\mathsf{PHF}(5; 405, 181, 6)$. Then by Theorem 4.2, the existence of a $\mathsf{CA}(N; 6, 181, v)$ ensures that a $\mathsf{CA}(5N - 5; 6, 405, v)$ exists. This yields the smallest covering array for these parameters when $v \in \{5, 7, 8, 9, 11, 13, 18, 19\}$.

Extending the Blackburn method to fractal and heterogeneous hash families therefore improves on known constructions for covering arrays even within the ranges currently tabulated at [24]. To see that the extension to distributing hash families is also effective, we consider larger strengths. We use the framework of Lemma 4.16, taking $\kappa = 10$. According to [25], a $\mathsf{PHF}(4; 10, 6, 4)$ exists, and it can be easily verified that one is fractal. Then a $\mathsf{PHHF}(7; 70, 54, 9)$, and hence a $\mathsf{DHHF}(7; 70, 54, 9, p)$, exists for all $2 \leq p \leq 9$. Using instead the DHF(4;10,4,4,2) from Table 4.1 in the construction of Lemma 4.16, we produce a $\mathsf{DHF}(7;70,46,9,2)$, using many fewer symbols. When used in a column replacement strategy for covering arrays, this enables us to use a binary covering array with 46 columns rather than 54, which can be a substantial improvement.

4.6 Conclusion

We have developed a new recursive technique to generate heterogeneous hash families from the use of enforcing a structural constraint on the ingredient hash family; namely, that it is fractal. From this, we investigated many types of coverings, and have improved many parameters for perfect hash families. Improvements to distributing hash families have also improved on the sizes of covering arrays.

Chapter 5

GENETIC ALGORITHMS FOR TRANSFORMATIONS OF EXISTENTIAL RESTRICTIONS

In recent work by Colbourn and Lanus [33], efficient algorithms to construct covering perfect hash families were considered. Specifically, an array with k columns was given, and suppose that mk columns for a CPHF are desired. One technique to achieve this goal is to proceed with two stages: first, horizontally juxtapose the CPHF m times to achieve an initial array on mk columns; and second, use some algorithm to "complete" the array. The second step is necessary because unless the copies of the initial CPHF are modified, any t-set of the mk columns that involve at least two that correspond to the same initial column cannot be separated in any row. Note that the number of t-sets of columns not separated is known precisely here, and completely determined; however, this number is a significant fraction of the total number of t-sets.

In an attempt to mitigate this problem, the authors considered applying *affine transformations* to each row and to each of the copies, which corresponds to modifying each entry by multiplying it by an adder and a multiplier, and performing the arithmetic in $\mathbb{F}_v$. The key here is that for any t-set that is contained entirely within a single copy, it is always separated because affine transformations are applied. For CPHFs, the sample space is very large, so Colbourn and Lanus used a greedy algorithm for finding appropriate transformations. For each row r of the CPHF, determine some (or all) of the possible affine transformations that contain two identical columns from two blocks, if it is to be applied to r. Then they choose any transformation that yields the smallest number of t-sets of components not fully separated. If there is a tie, choose the lexicographically first transformation. For CPHFs, it turns out that affine transformations are exactly the operations that preserve the structure of each CPHF copy.

We extend their work for a general class of t-restrictions, and to use genetic algorithms to find the best "transformations". We also generalize the notion of affine transformations to *t-transformations* of the underlying hypergraph involving the t-sets. The goal of these algorithms is to maximize the "fitness" of the corresponding t-restriction, so that deterministic methods can "complete" the array. We then report computational results for maximizing the fitness for PHFs, as they have many transformations compared to the number of symbols allowed for it. Some preliminary work of this chapter appears in [47].

5.1 Prior Work

We first illustrate the general framework of genetic algorithms. In such an algorithm, there is a population P of *individuals*, wherein each individual has an associated *fitness* according to some *fitness function* f. Typically, one wants to maximize the average fitness of all individuals in P. First, an initial population P_0 of N individuals is created, usually "at random." Then, as long as some fitness criterion is not satisfied, changes to the population are applied, also usually at random; suppose the current population is P_i. We desire to form population P_{i+1} with average fitness at least that of P_i.

Initially, $P_{i+1} = P_i$. A genetic algorithm often uses *tournament selection*, which seeks to pick "fit" individuals in the population. Specifically, it randomly chooses a subset of the population of a given size, and any individual with highest fitness in the subset is selected (if there are ties, break them arbitrarily). Here, we run tournament selection until two distinct individuals A, B are selected, and removed from

P_{i+1}. The *crossover* operator forms two *offspring* o_1, o_2 based on A, B by combining different properties of them. Then, one performs *mutation* on the two children; usually this corresponds to modifying each individual's attributes. Next, the fitness value is calculated for both children according to f. And finally, the two fittest individuals of the four individuals–the two parents and the two offspring–are selected to be inserted into P_{i+1}. If there is a tie among the fitnesses, we prefer to select the two (mutated) children, simply for an attempt to exit local optima (i.e., random single changes to the population may possibly not improve the overall population's fitness). This "steady-state" algorithm is given in Algorithm 3; note that since the two fittest individuals are always inserted into P_{i+1}, the average fitness never decreases.

Algorithm 3 General Steady-State Algorithm

1: Let P_0 be a population generated at random, and $i = 0$.
2: **while** condition to stop has not been satisfied **do**
3:     Set $P_{i+1} \leftarrow P_i$.
4:     Run tournament selection on P_i to find two individuals I_1, I_2.
5:     Crossover I_1, I_2 to obtain two children C_1, C_2, with probability $p_{crossover}$. (If not performed, skip Steps 6–8, and re-insert I_1, I_2 back into P_{i+1}.)
6:     Mutate the two children to obtain M_1, M_2, independently with probability p_{mutate}.
7:     Pick the two fittest individuals F_1, F_2 from $\{I_1, I_2, M_1, M_2\}$. (if there are ties, prefer M_1, M_2 over I_1, I_2)
8:     Insert F_1, F_2 into P_{i+1}.
9:     $i \leftarrow i + 1$.
10: **end while**

Most prior work for applying metaheuristic techniques to t-restrictions were to covering arrays, and only to a limited extent. Genetic algorithms were first used for covering arrays by Stardom [73]. More recently, such techniques that exploited the search space were performed by Rodriguez-Tello and Torres-Jimenez [64], and even more recently by Sabharwal et al. [65]. Other metaheuristic techniques have been extensively used on covering arrays, such as simulated annealing, tabu search, ant colony optimization, and particle swarm optimization. See [77] for an extensive survey on such techniques.

Even though the techniques used in all prior work were not all the same, they have a common framework: the representation of each individual was the array itself, and random mutations/operators to the array were formed until it was a covering array (or an iteration limit was reached). In the case of genetic algorithms, there are two operators: crossover, and mutation. Crossover involves switching either entire rows, entire columns, or just single elements of the two parents; and mutation involves randomly changing one or more values in the array. In all cases, the fitness of an individual is the number of t-way interactions, in the case of covering arrays. As far as we are aware, there is no previous work on metaheuristic algorithms for hash families, nor most other commonly used t-restrictions. Later in this chapter, we present a method that is more efficient at generating PHFs using a genetic algorithm.

5.1.1 A Genetic Algorithm for PHFs Based on Prior Work

Because there has been no work on PHFs with a GA previously, we outline our representation that is inspired by previous work with covering arrays. An individual is an $N \times k$ array on v symbols, and the fitness function is the number of t-sets that are λ-separated, as expected. Here, we consider homogeneous PHFs, but the method is easily extensible to homogeneous ones.

The mutation operator is slightly different than expected: a row and a column are selected uniformly at random. However, many runs of the algorithm yielded little increases in the average fitness if one selects a random value to be placed into this entry, even after many generations. Instead, we deterministically choose a value such that setting the row/column of the array to this value (and keeping the rest of the array fixed) yields the highest fitness, breaking ties based on a fixed ordering of the symbols. This way, not only does the fitness not decrease purely based on mutation alone, but the population is more diverse because often two different symbols yield the same largest fitness, and both are often chosen.

The crossover operator used here is *one-point crossover*, in that when two parents are crossed, one selects a random point in their representation to yield two partitions. In the first part, the child has the same representation as the first parent, and from the second parent for the second part. Because PHFs are based on separation within rows only, we naturally select a partition of the rows for what the child receives from its parents.

5.2 A Genetic Algorithm for transformations for Existential t-Restrictions

We extend previous uses of genetic algorithms to work for arbitrary t-restrictions. First, we need several definitions. Let $A_{N,k,\mathcal{T}}$ be the set of all $N \times k$ arrays that satisfy the t-restriction $\mathcal{T}$. A *t-transformation for N rows and k columns* is a bijective function $\phi_{N,k,\mathcal{T}} : A_{N,k,\mathcal{T}} \to A_{N,k,\mathcal{T}}$ (when $N, k, \mathcal{T}$ are implied, we drop them from the notation). In other words, A is an array satisfying a t-restriction $\mathcal{T}$ if and only if $\phi_{N,k,\mathcal{T}}(A)$ also satisfies the same restriction.

However, we focus on a certain type of t-restriction, which allows for the t-transformations chosen to have useful properties. A t-restriction where all of the quantifiers are $\exists$ is called an existential t-restriction. All hash families that have been discussed in this thesis are existential t-restrictions. We focus on t-transformations for existential t-restrictions such that one can rewrite the transformation as a composition of N transformations $T_1, \cdots, T_N$ such that T_i only modifies the entries in row i. Hash families fit under this type of transformation, since the separation condition is only on a per-row basis.

We give several examples. Colbourn and Lanus [33] showed that if $\phi(x)$ is of the form $ax + b$ where $a, b \in \mathbb{F}_q$, and the arithmetic is performed in $\mathbb{F}_q$, then ϕ is an transformation when the array A is a SCPHF. For perfect hash families (and many of their generalizations other than SCPHFs), then one can take ϕ to be any permutation of the symbols because if $v_1, \cdots, v_t \in V$ are all distinct, then since ϕ is a permutation and hence is bijective, then $\phi(v_1), \cdots, \phi(v_t)$ are also all distinct.

This observation leads to a more useful genetic algorithm than previous approaches. We first describe the individual representation, and then a high-level explanation of its effectiveness. Suppose that

- $\mathcal{T}$ is an existential t-restriction,
- k is the desired number of columns,
- we are given an array A with $c < k$ columns and N rows satisfying $\mathcal{T}$, and
- each row i of A can use (at most) v_i symbols.

For any d such that $t \leq d \leq c$, we define a set of arrays $\mathsf{B}_1, \cdots, \mathsf{B}_m$ all with d columns a *d-subpopulation*; also, we say that these arrays are *within a d-subpopulation*. The representation of a single individual I with N rows, where I is within a d-subpopulation, contains the following attributes:

- a list of $c - d$ columns $C_1, \cdots, C_{c-d}$ (that are to be deleted); and
- let m be the smallest integer such that $dm \geq k$. Then I contains $N(m-1)$ transformations $\phi_{1,1}, \cdots, \phi_{N,m-1}$. (These correspond to each of the rows of all copies, not the original array)

The entire population $\mathcal{P}$ consists of a specified number x_d of individuals in a d-subpopulation for each d with $t \leq d \leq c$. The fitness of an individual I in a d-subpopulation is determined as in Algorithm 4.

Algorithm 4 Calculating the Fitness of an Individual I within a d-subpopulation

1: Remove the given $c - d$ columns $C_1, \cdots, C_{c-d}$ from A (according to I) to obtain an array A_1.

2: Horizontally duplicate A_1 $m - 1$ times to obtain an $N \times dm$ array A_2.

3: Apply the corresponding t-transformations (according to I) to each of the $m - 1$ copies to obtain array A_3.

4: Determine, among all t-sets of columns with at least one duplicate modulo c, how many of them are λ-separated.

5: While there are strictly more than k columns (the target) in A_3, remove any column participating in the largest number of column sets that are not λ-separated; let the final array be F.

6: Return the number of t-sets of columns that are λ-separated in F.

We give a high-level intuition for why this GA setup is more useful for constructing arrays than the standard method, and why d-subpopulations (along with removing $c - d$ columns to obtain an array within one) are helpful in finding the best arrays. Not only is the fitness function more efficient to calculate (since any d-subpopulation has the exact same t-sets to check, and there are fewer than all possible t-sets), but each individual has a smaller representation than the entire array, given that the number of columns is sufficiently large.

There are two further advantages. Suppose we want to apply affine transformations to PHFs, like the method of Colbourn and Lanus. If a row has v symbols, there are only $v(v - 1)$ possible choices of transformations to select (multiplier being nonzero, and adder being any value), whereas there are $v!$ possible t-transformations for PHFs, which allows for much greater possibility in generating "good" copies of the original PHF. The other large benefit here is the deletion of the $c-d$ columns; by choosing d to be larger, then the number of copies needed to reach a given target of columns is larger, which allows for a more effective "spread" within the set of all transformations.

The mutation operator is natural given the setup: if ϕ is the observed transformation and it is to be mutated, we update it to be a different t-transformation (we assume that the corresponding set of t-transformations has at least two elements); and if we are to mutate the $c-d$ columns to delete, then the operator chooses a different set of columns.

For crossover, since the $c-d$ columns to delete are unordered (since it does not matter what order they appear), we take the union of the $c-d$ columns of both parents, and the child then takes a random subset of size $c-d$ where at least one column comes from both parents. However, we use a different approach with regards to the crossover operator for the t-transformations, developed by Davis [41], and is reproduced here for convenience. Suppose that ϕ_1, ϕ_2 are two t-transformations that are to be crossed and to form offspring. Further suppose that the considered row allows for v possible symbols, $\phi_1 = (x_1, \cdots, x_v)$, and $\phi_2 = (y_1, \cdots, y_v)$ (this notation indicates that $\phi_1(i) = x_i$ and $\phi_2(i) = y_i$). To form an offspring O (initially with undetermined values ∞), we choose a randomly chosen subsequence $(x_i, \cdots, x_j)$ from ϕ_1, and copy these entries element-wise into O; at this point, O is of the form $(\infty, \cdots, \infty, x_i, \cdots, x_j, \infty, \cdots, \infty)$. To form the other elements for O, consider all of the other elements in ϕ_2 after removing $(x_i, \cdots, x_j)$. Now insert them into O's undetermined entries in their proper order according to ϕ_2. This procedure works because ϕ_1, ϕ_2 are permutations of each other. We give an example. Suppose that $\phi_1 = (0, 2, 4, 3, 1)$ and $\phi_2 = (4, 3, 2, 1, 0)$. We pick the subsequence $(2, 4, 3)$ from ϕ_1, and the offspring O

is initially $O = (\infty, 2, 4, 3, \infty)$. Removing these three chosen elements from ϕ_2 yields the subsequence $(1, 0)$, so we insert these two elements in this order into O to form $(1, 2, 4, 3, 0)$.

By far the most computationally intensive part of the algorithm is the fitness function, determining the number of t-sets left unseparated. One improvement that can be made immediately is to only consider t-sets $S = \{c_1, \cdots, c_t\}$ such that C involves at least two of the copies (because otherwise, C is separated by the definition of the considered t-transformations). Further, any individual that is not mutated (i.e., the original parents) do not need to have their fitness function re-evaluated. We use these improvements in our implementation of Algorithm 4.

Our genetic algorithm does not change which subpopulation an individual is in, for the reason that the optimal choices for d columns to delete may not at all be similar to those for $d' \neq d$ columns deleted. For this reason, we run our algorithm for each choice of subpopulation, because each of them can be run independently. An interesting research direction here is to find a useful crossover operator between two individuals that live within different subpopulations.

5.3 Genetic Algorithm Computational Results

The goal of our GAs is not necessarily to generate t-restrictions, but to generate arrays that are as close as possible to being t-restrictions. Nevertheless, our GA can and does improve on the best-known values of PHFN_λ. Atici [9] showed that $\mathsf{PHFN}_1(x, 4, 3) = 4$ for all $9 \leq x \leq 16$. Therefore, $\mathsf{PHFN}_3(x, 4, 3) \leq 12$ for the same range. Here, we show that $\mathsf{PHFN}_3(14, 4, 3) \leq 11$, given in Figure 5.2. The starter array was a $\mathsf{PHF}_3(11; 9, 4, 3)$, found via the satisfiability method in Section 3.2.3, given in Figure 5.1. The GA parameters involved deleting 2 columns ($d - c = 2$), with 2 parts ($m = 2$), 100 individuals, and the resulting PHF_3 was found within 88 iterations of the

$$\begin{array}{ccccccc}
2 & 3 & 3 & 1 & 2 & 1 & 3 \\
0 & 0 & 3 & 3 & 1 & 0 & 1 \\
0 & 3 & 1 & 3 & 1 & 1 & 1 \\
2 & 1 & 3 & 2 & 3 & 1 & 1 \\
0 & 3 & 2 & 3 & 1 & 0 & 3 \\
0 & 3 & 3 & 2 & 1 & 3 & 2 \\
1 & 3 & 3 & 0 & 1 & 2 & 1 \\
2 & 1 & 2 & 1 & 1 & 3 & 2 \\
0 & 0 & 0 & 3 & 2 & 2 & 1 \\
0 & 0 & 3 & 1 & 2 & 1 & 1 \\
1 & 0 & 1 & 3 & 1 & 1 & 3
\end{array}$$

Figure 5.1: A $\mathsf{PHF}_3(11; 9, 4, 3)$.

`while` loop. The run-time of the algorithm was less than 20 seconds (combined with initially generating the starter array $\mathsf{PHF}_3(11; 9, 4, 3)$ with the satisfiability solver), whereas applying the solver directly to generating a $\mathsf{PHF}_3(11; 14, 4, 3)$ took over 2 minutes.

We showcase attempting to generate a $\mathsf{PHF}_2(12; 50, 4, 3)$ using our GA to demonstrate its effectiveness, and compare it to using previous techniques where the representation of the individual is the entire array. The "starter" array is a $\mathsf{PHF}_2(12; 30, 4, 3)$, displayed in Figure 5.3. It was generated using the SAT model developed earlier in Section 3.2.3. We picked 30 columns because (1) it is strictly more than half of the number of desired columns (which forces $c - d > 0$), and (2) the SAT representation for this array is very large. Furthermore, we picked 4 symbols because the set of t-transformations for the symbol set is larger than the number of affine transformations, and so there is a potential advantage over the old approach. The choice of $\lambda = 2$

$$\begin{array}{c}
2\ 3\ 3\ 1\ 2\ 1\ 3\ 3\ 0\ 0\ 2\ 3\ 2\ 0 \\
0\ 0\ 3\ 3\ 1\ 0\ 1\ 2\ 2\ 3\ 3\ 1\ 2\ 1 \\
0\ 3\ 1\ 3\ 1\ 1\ 1\ 0\ 3\ 2\ 3\ 2\ 2\ 2 \\
2\ 1\ 3\ 2\ 3\ 1\ 1\ 2\ 0\ 3\ 2\ 3\ 0\ 0 \\
0\ 3\ 2\ 3\ 1\ 0\ 3\ 0\ 2\ 3\ 2\ 1\ 0\ 2 \\
0\ 3\ 3\ 2\ 1\ 3\ 2\ 1\ 0\ 0\ 2\ 3\ 0\ 2 \\
1\ 3\ 3\ 0\ 1\ 2\ 1\ 0\ 1\ 1\ 2\ 0\ 3\ 0 \\
2\ 1\ 2\ 1\ 1\ 3\ 2\ 3\ 0\ 3\ 0\ 0\ 2\ 3 \\
0\ 0\ 0\ 3\ 2\ 2\ 1\ 2\ 2\ 2\ 1\ 3\ 3\ 0 \\
0\ 0\ 3\ 1\ 2\ 1\ 1\ 3\ 3\ 2\ 1\ 0\ 1\ 1 \\
1\ 0\ 1\ 3\ 1\ 1\ 3\ 2\ 3\ 2\ 0\ 2\ 2\ 0
\end{array}$$

Figure 5.2: A $\mathsf{PHF}_3(11; 14, 4, 3)$.

$$\begin{array}{c}
012001022113223300130313102223 \\
000111122230011021223310031213 \\
002311210302301310100121021313 \\
011222203320113300020333102231 \\
010123312030021210302223321303 \\
001201222033331002032020123112 \\
002122310322022131112110300303 \\
002101231231032312213100112033 \\
000113112003011101323223212131 \\
012023320223321131223330223033 \\
011303103003133122332233211010 \\
011132100303010033102102003212
\end{array}$$

Figure 5.3: A $\mathsf{PHF}_2(12; 30, 4, 3)$.

was to showcase the algorithm's applicability to higher index restrictions. We added columns until the solving time for the SAT solver took over an hour. It is unknown whether a $\mathsf{PHF}_2(12; 50, 4, 3)$ exists, but our goal here is to maximize the fitness of a 12×50 array.

We now give two examples of attempting to generate PHFs with larger parameters. Figures 5.4 and 5.5 display the results of applying our GA to generating a $\mathsf{PHF}_2(12; 30, 4, 3)$. In all experiments, we used 1000 iterations, and 100 individuals. The maximum theoretically possible fitness for any individual is $\binom{30}{3} = 19600$. The `Standard` plot consists of the standard representation discussed earlier, and `2 Copies`, `3 Copies`, and `7 Copies` correspond to the number of copies formed. In each case, we determined the minimum number of columns to remove so that the generated PHF has more than the necessary columns; for 2 copies, this was 5 columns deleted; 3 copies had 13 columns; and 7 copies had 17 columns. Each of the plots shows the average of the corresponding heuristic over 10 runs of the GA: Figure 5.4 is the average maximum fitness, and Figure 5.5 is the average mean fitness.

Note that the fitness of the individuals, when initialized randomly, is very high, above 90% for nearly all individuals in all runs. This will often be the case when the number of rows is large enough, and when the index corresponding to the fitness function matches that of the starter array.

When the strength is larger, this threshold for the number of rows is larger; for this reason, we consider an example with few rows and high strength, namely strength 6. A $\mathsf{PHF}_1(4; 7, 6, 6)$ is given in Figure 5.6, and is optimal in the number of rows and columns. We desire to construct an array that is as close as possible to being a $\mathsf{PHF}_1(4; 14, 6, 6)$. Figures 5.7 and 5.8 display the results of applying our GA to generating a $\mathsf{PHF}_1(4; 14, 6, 6)$.

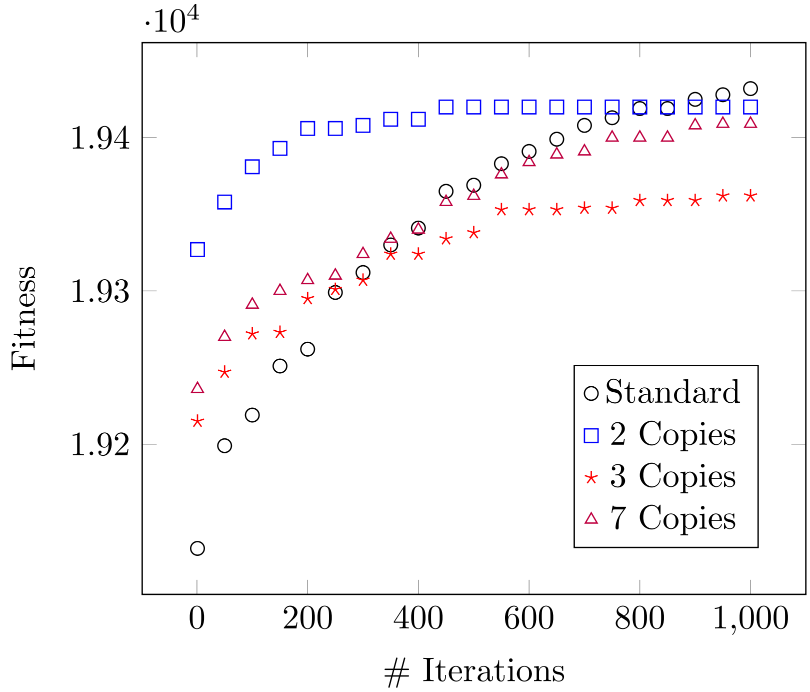

Figure 5.4: Scatter Plot for Generating a $\mathsf{PHF}_2(12; 50, 4, 3)$. Values Shown Are the Maximum Fitnesses over All Individuals, Taken over 1000 Iterations, Averaged over 10 Runs of the Algorithm.

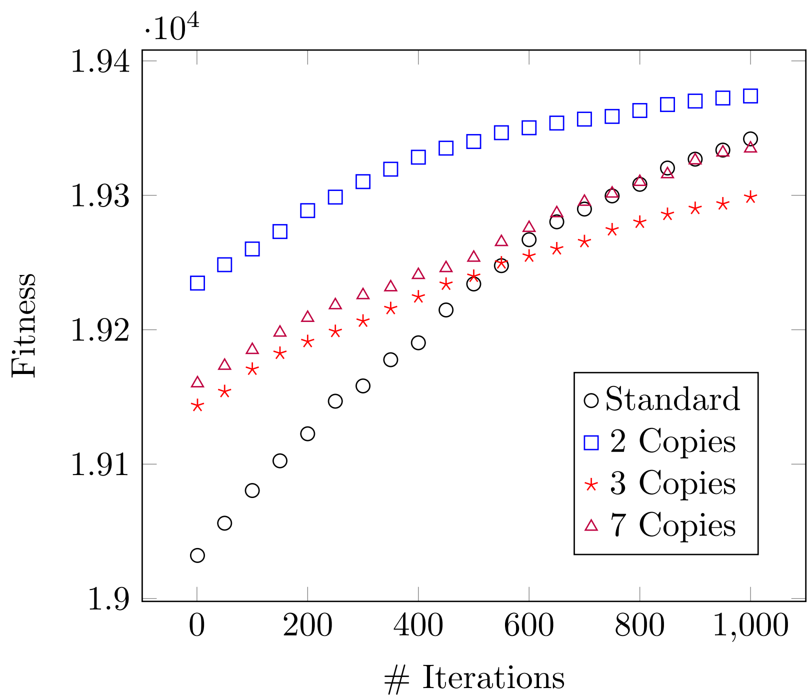

Figure 5.5: Scatter Plot for Generating a $\mathsf{PHF}_2(12; 50, 4, 3)$. Values Shown Are the Average Fitnesses over All Individuals, Taken over 1000 Iterations, Averaged over 10 Runs of the Algorithm.

0 1 2 3 4 5 0

0 2 3 4 5 2 1

0 1 4 5 4 2 3

0 1 2 5 3 4 5

Figure 5.6: A $\mathsf{PHF}_1(4; 7, 6, 6)$.

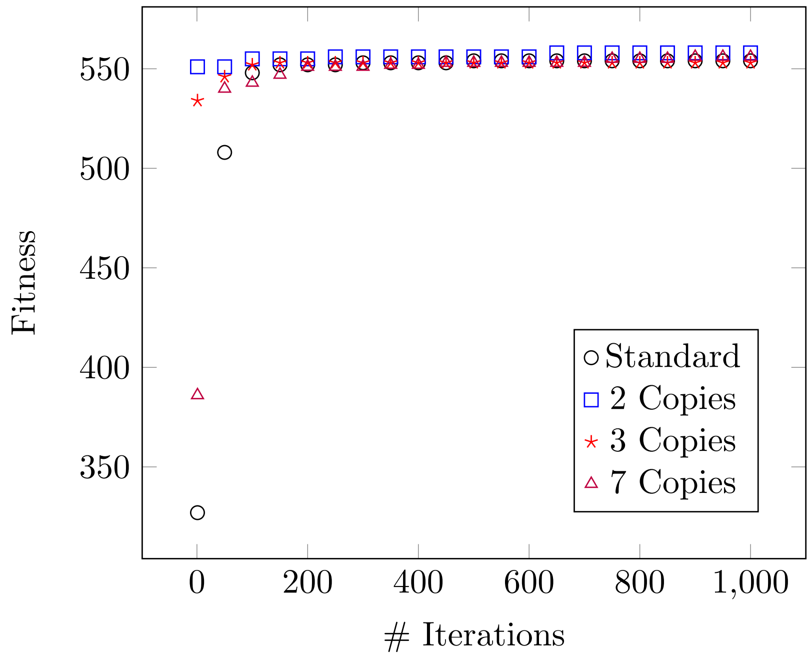

Figure 5.7: Scatter Plot for Generating a $\mathsf{PHF}_1(4; 14, 6, 6)$. Values Shown Are the Maximum Fitnesses over All Individuals, Taken over 1000 Iterations, Averaged over 10 Runs of the Algorithm.

5.3.1 Discussion of GA Results

We discuss the results of the last two experiments; we begin with the first example, generating a $\mathsf{PHF}_2(12; 30, 4, 3)$. The figures showcase results that are in agreement with our hypothesis, in that using copies to generate the object not only produces highly fit individuals quickly, but also these individuals are more fit (at first) compared to usual techniques (`Standard` in the plots). However, we can see that `Standard` overtakes all three choices of copies in Figure 5.4, and all but `2 Copies` in Figure 5.5.

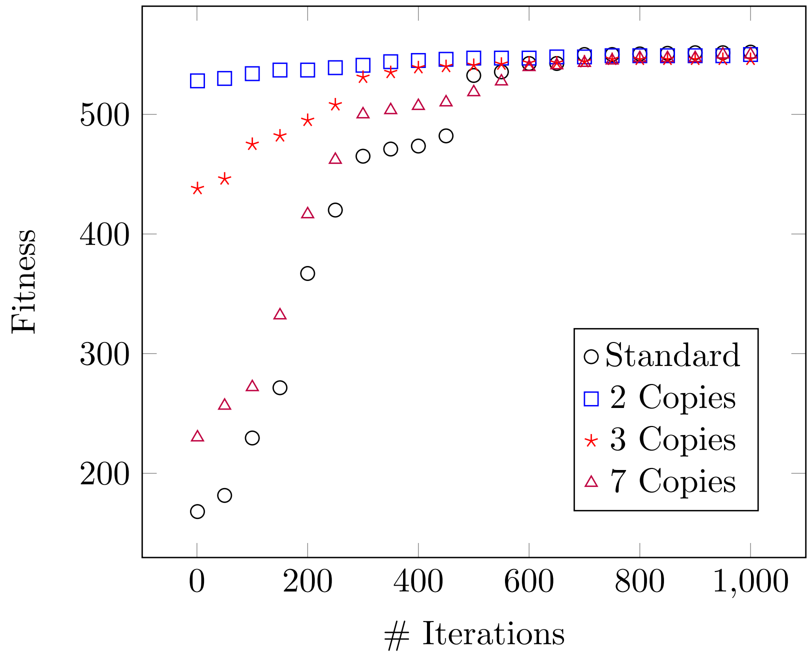

Figure 5.8: Scatter Plot for Generating a $\mathsf{PHF}_1(4; 14, 6, 6)$. Values Shown Are the Average Fitnesses over All Individuals, Taken over 1000 Iterations, Averaged over 10 Runs of the Algorithm.

Intuitively, the GA for `Standard` can only make a small number of changes to the array, which can lead to local optima. However, because `Standard` chooses a random number of entries to mutate between 1 and 10, it is able to escape local optima relatively well. It overtakes the other plots because each of them always mutates exactly the same number of resulting entries in the generated PHF every time. We decided not to have their GAs mutate multiple t-transformations simultaneously because it would then mutate a significantly larger portion of the array, whereas `Standard` only mutates a small portion.

Now to the second example, generating a $\mathsf{PHF}_1(4; 14, 6, 6)$. Such an array does not exist, because $\mathsf{PHFN}_1(14, 6, 6) \geq 17$ due to a known lower bound [50]. The results are similar as that of the other example, but give slightly different conclusions. Here, Figure 5.7 shows that all choices of copies has the fitness converge very quickly. Note that in this example for n copies, there are $(6!)^n$ choices of t-transformations, and for the first example, there are $(4!)^n$, which is much larger, even for $n = 2$. Also take note of Figure 5.8, which shows that `7 Copies` has the lowest fitness among all four plots other than `Standard`. Here, 7 copies corresponds to deleting 5 columns, resulting in two columns for each copy. We predict this is one of the reasons why the plot is "slow" to converge, because $t = 6$, which is larger than the number of columns for each copy. In the first example, each number of copies has that each one has more columns than the strength. This shows that even though it is not as fast to converge, our method works for all choices of copies and strengths.

For both examples, the run time of the GA was much faster for `2 Copies`, `3 Copies`, and `7 Copies` compared to `Standard`; specifically, all of them achieved a run time which is substantially less than half than for `Standard`. This is one additional benefit of our approach, as suspected, in that the number of t-sets needed to be checked is much smaller.

Finally, we address the two measures that we considered: average and maximum fitness. Note that in most real-world applications of PHFs, only the fact that it is a PHF matters, and not necessarily the structure within it. Therefore, we are only concerned with finding PHFs of maximum fitness. However, some applications may require "diverse" individuals, but would want many of the individuals to still have large fitness. This motivates the study of having many "different" PHFs of high fitness. Our approach does not aim to find many individuals with high fitness; the goal is purely to find any individual with as high of fitness as possible.

That said, it is possible to use an algorithm such as NSGA-II [42], which is useful for multi-objective optimization. For PHFs, fitness would be an obvious objective to maximize, but some other objectives that are worth considering are (1) the spectrum of t-sets separated [1] , (2) the balance of symbols for each row, (3) average number of

[1]This is a mapping between t-sets and the index of the λth row that separates each one.

times each t-set is separated, among others. We anticipate the use of multi-objective optimization for many types of t-restrictions.

5.4 Conclusion

In this chapter, we developed a genetic algorithm that aims to construct existential t-restrictions of arbitrary index, given a "starter array" with fewer columns than what the target is. A key component of this algorithm is that it uses t-transformations of the given t-restriction, and multiple copies of the starter array. Further, different t-transformations can be chosen for each row of each copy. The insight in this algorithm is that although the number of t-sets separated in each row is the same (by the t-transformation property), *where* the t-sets are separated changes. Before, if a t-set of columns is not separated in some row, then it is not separated in any row; with our approach, this does not happen nearly as often in practice.

Chapter 6

CONCLUSIONS

In this chapter we summarize all of the results and ideas presented in this thesis. We then give open problems as well as research directions that are worth exploring.

6.1 Main Results and Ideas

The main contribution of this thesis is a greater insight into the structure and generation of perfect hash families of higher index, through the use of probabilistics and asymptotics, constructive algorithms, new recursive constructions, and genetic algorithms. We enumerate individual contributions of each chapter next.

In Chapter 3, we investigated hash families of higher index. We gave simple constructions of them, mainly inspired by methods from coding theory. We then gave a new recursive construction that not only exploited ingredient hash families with smaller index, but also improves on the sizes of many PHFs with index 1, even with small strength. More importantly, this construction works with any number of columns, symbols, and strength, and is general enough to allow for improvements when the number of columns exceeds the strength. We analyzed PHFN_λ both in a probabilistic setting (for which the optimal size is met for all choices of columns, symbols, and strength), and asymptotically. Finally, we developed a conditional expectation algorithm that constructs PHF_λs that meet these bounds. The contribution of this chapter is a greater understanding of the structure of higher-index perfect hash families, both computationally and asymptotically.

In Chapter 4, we developed a new recursive construction algorithm for PHFs when the number of rows is small, and the number of symbols is relatively high (namely, when the optimal number of columns is linear in the number of symbols). Here, we generalized the notion of PHFs to incorporate more requirements of separation, which we called fractal. Fractal PHHFs were developed using simple constructions, but they yielded many improvements in the sizes of PHFs. The contribution of this chapter is a new constraint on hash families wherein a new recursive construction is developed, and improving on the sizes of many perfect hash families.

In Chapter 5, we extended a method of Colbourn and Lanus to generate PHFs using a genetic algorithm. Furthermore, the genetic algorithm is novel because it (1) exploits the structure of PHFs, (2) guarantees substructure given an ingredient PHF, (3) improves the run time for computing the fitness of the PHF, and (4) is able to be generalized to any existential t-restriction. The contribution of this chapter is a new (randomized) construction algorithm that finds arrays with very high fitness quickly, whereas other methods are slower or do not produce arrays with high fitness.

6.2 Future Research Directions

In this section we give some of the many possible open problems and research directions that emerge from the topics presented within this thesis.

6.2.1 Higher Index Research Directions

It is an open problem, even in the $\lambda = 1$ case, whether or not there exists a satisfiability formula for perfect hash families that is polynomial in N, k, v, t, λ (the one presented in Chapter 3 is of exponential size). If v, t, λ are fixed, then the size of the array is polynomial in the values of N, k. There are alternate formulations of satisfiability formulas for covering arrays other than the standard one [52], but they do not appear to be applicable to perfect hash families, primarily because of the discussion in Chapter 3.

What better purely probabilistic bounds, both lower and upper, can be found for PHFN_λ? The existing techniques do not appear to be able to improve beyond the additive $c_2\sqrt{\lambda \log k}$ term. We believe a better bound of the form $c_1 \log k + c_2 f(\lambda) \log\log k$ can be found via probabilistic techniques, where c_2 is not dependent on λ or k, and the function f is only dependent on λ. Here is our reasoning, and we explain why it falls short; we illustrate this with $\lambda = 2$. Generate a PHF_1 uniformly at random, and suppose it has N rows. Then according to the binomial distribution one would hope to be able to bound the number of t-sets separated once, but not twice. Using a variable-strength analogue of perfect hash families [60], one can bound the number of additional rows needed (which is essentially logarithmic in the number of t-sets left only 1-separated). The issue with this reasoning is that just because the expected number of sets separated 0 times is strictly less than 1 does not imply much about the distribution for sets separated exactly once. If one can employ probabilistic tools to infer what the distribution is, then we conjecture that a bound of the above form can be obtained somewhat simply.

For probabilistic bounds, we state a research direction with regards to the Lovász Local Lemma. It is possible to setup the dependency graph to apply the *asymmetric* LLL for a bound on PHFN_λ, stated next.

Theorem 6.1. *[7] Let $E_1, \cdots, E_n$ be events in a probability space, and let $\Gamma(E_i)$ be the set of neighbors of E_i in the dependency graph. In the dependency graph, event E_i is not adjacent to events which are mutually independent to it. If there exist $0 \leq x(E_1), \cdots, x(E_n) \leq 1$ such that $\Pr[E_i] \leq x(E_i) \prod_{E_j \in \Gamma(E_i)} (1 - x(E_j))$, then with nonzero probability all of $E_1, \cdots, E_n$ simultaneously do not occur.*

The issue in applying the symmetric LLL directly here is that when a set C is i-separated, the event in which it is $(i+1)$-separated depends on the former event, and not the other direction. One can increase the dependence by adding a directed edge in the other direction, but this would not achieve as strong a bound. We outline how to construct the graph: there are $\binom{k}{t}\lambda$ vertices $A_{C,i}$, to denote the event that C is exactly i-separated, for all $C \in \binom{[k]}{t}$ and $0 \leq i < \lambda$. Construct a directed edge from $A_{C,i}$ to $A_{C',i'}$ if:

- $C \neq C'$, and $C \cap C' \neq \emptyset$;
- $C = C'$, and $i' = i + 1$.

Effectively, t-sets that normally have dependence appear in the graph, and a directed path exists among the same t-set in terms of increasing index. It is an open question as to what bound can be obtained by considering this dependency graph, since the result of Deng, Stinson, and Wei depends on the fact that all vertices have the same degree, and the entire graph is symmetric.

What are better asymptotics on PHFN_λ, where λ can grow? Additionally, what if v, t also can grow? The key here appears to be finding an asymptotically equal expression for the failure probability.

6.2.2 *Fractal Research Directions*

A research direction for fractal PHHFs, naturally, is finding better constructions for higher-index hash families, and more generally to separating heterogeneous hash families. We give a setup for separating hash families here. For a set $W = \{w_1, \cdots, w_s\}$, the *shadow* of W is the union of $\{w_1 - i_1, \cdots, w_s - i_s\}$ over all choices of $i_1, \cdots, i_s$ such that exactly one of the i_j is 1 and the others are 0 (if any of the i_j become 0 as a result, then we drop it from the notation). For example, if $W = \{2, 4\}$, the shadow of W is $\{\{1, 4\}, \{2, 3\}\}$. The *recursive shadow* of W is the union of the shadow S of W as well as the recursive shadow of every $s \in S$. If $W = \{2, 4\}$, then the recursive shadow of W is $\{\{1,4\},\{2,3\},\{1,3\},\{1,2\},\{1,1\},\{1\},\{2,2\}\}$. Construct a tree T_W rooted at W, with vertices being the sets in the recursive shadow of W as well as W, and an edge formed between s, s' if s is in the shadow of s', or vice versa. The *recursive shadow height* of T_W is the length of the longest root-to-leaf path in T_W from W. The *shadow* of W is the union of all shadows of every $W \in \mathsf{W}$; define recursive shadow, and recursive shadow height similarly for W. Denote the recursive shadow height of W as $r(\mathsf{W}) = \max_{W \in \mathsf{W}} r(W)$.

A $\mathsf{SHHF}_\lambda(r(\mathsf{W}) + \lambda - 1; k, (v_1, \cdots, v_N), \mathsf{W})$ is *fractal* if and only if the removal of any row i results in a fractal $\mathsf{SHHF}_\lambda(r(\mathsf{W}) + \lambda - 1; k, (v_1, \cdots, v_{i-1}, v_{i+1}, \cdots, v_N), \mathsf{W}')$, where W' is the shadow of W'. One can now prove similar results as presented in Chapter 4. However, it would be of great interest to determine when and if the generalization to SHHFs generates arrays that are substantially better than existing methods for constructing SHHFs.

Another research direction here is how to generate the (n, m, d, λ) ingredients, or to find "good" constructions of them that are stronger than the binomial result of Lemma 4.10. Further, how do computational results compare to the ones generated here? The key with (n, m, d, λ) coverings for all choices of λ is that not all of the n subsets chosen are not required to have the same cardinality.

Every fractal PHF with $N = t = 3$ that we have generated matched the best known values of k for each number of symbols v. In other words, for every best-known PHF that exists with 3 rows and strength 3, we found a fractal PHF with the same parameters. This observation leads to the question of whether the optimal parameters for fractal PHF always match those of "non-fractal" PHFs. A simple probabilistic analysis of the $N = t = 3$ case gives heuristics as to why this is the case; find the maximum number of columns for a 1-separated PHF on 3 rows and strength 3, and the same for a 2-separated PHF on 3 rows and strength 2, and observe the minimum of the two. If one shows that the minimum asymptotically matches the maximum of the two, then there is evidence that this is true. However, this does not imply that a fractal PHF on the minimum of the two actually exists because the distribution may be completely different when both requirements are considered.

6.2.3 Genetic Algorithm Research Directions

In the genetic algorithm we proposed in Chapter 5, we used a steady-state approach. When one considers PHFs generated at random, the fitness of them is often very large; for the example presented there, the fitness of a random individual is over 90%. Most genetic algorithms are suited for problems in which the fitness is often much smaller. Which genetic algorithm-style approach is best suited for problems where the average fitness is already very large (and the variance in the fitness distribution is very small), where the goal is only to maximize the fitness?

In unpublished work, we considered covering perfect hash families using a steady-state algorithm. There, it turned out that a mutation-only approach gave CPHFs that had much higher fitness, more quickly, than when mutation and crossover were both used. An intuitive explanation is that mutation is less of a "destructive" operator than crossover is. However, the crossover operator used in that work was not the one developed here, which is less "destructive" because a contiguous set of rows from each parent appears in the child. We implemented this crossover operator in our setting (t-transformations of PHFs), and similar results appeared. Would a similar crossover operator, or leaving crossover out entirely, improve the algorithm's fitness? Evidence seems to show that the latter is true, but since the fitness is already large on average, the first research direction of finding a more suitable GA may be more worthwhile.

REFERENCES

[1] Akhtar, Y. and F. K. H. Phoa, "A construction of cost-efficient designs with guaranteed repeated measurements on interaction effects", preprint (2019).

[2] Alon, N., "Explicit construction of exponential sized families of k-independent sets", Discrete Mathematics **58**, 2, 191–193 (1986).

[3] Alon, N. and S. Gutner, "Balanced hashing, color coding and approximate counting", in "Parameterized and Exact Computation", edited by J. Chen and F. V. Fomin, pp. 1–16 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009).

[4] Alon, N. and S. Gutner, "Balanced families of perfect hash functions and their applications", ACM Transactions on Algorithms **6**, 3, 54:1–54:12 (2010).

[5] Alon, N., D. Moshkovitz and S. Safra, "Algorithmic construction of sets for k-restrictions", ACM Transactions on Algorithms **2**, 153–177 (2006).

[6] Alon, N. and M. Naor, "Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions", Algorithmica **16**, 4, 434–449 (1996).

[7] Alon, N. and J. H. Spencer, *The probabilistic method* (John Wiley & Sons, 2004).

[8] Ang, N., *Generating Mixed-Level Covering Arrays with* $\lambda = 2$ *and Test Prioritization*, Master's thesis, Arizona State University (2015).

[9] Atici, M., *Hash families: recursive constructions and applications to cryptography*, PhD dissertation, University of Nebraska (1996).

[10] Atici, M., S. S. Magliveras, D. R. Stinson and W.-D. Wei, "Some recursive constructions for perfect hash families", Journal of Combinatorial Designs **4**, 5, 353–363 (1996).

[11] Barwick, S. G. and W.-A. Jackson, "A sequence approach to linear perfect hash families", Designs, Codes and Cryptography **45**, 1, 95–121 (2007).

[12] Barwick, S. G. and W.-A. Jackson, "Geometric constructions of optimal linear perfect hash families", Finite Fields and Their Applications **14**, 1, 1–13 (2008).

[13] Bazrafshan, M. and T. van Trung, "Bounds for separating hash families", Journal of Combinatorial Theory, Series A **118**, 3, 1129–1135 (2011).

[14] Blackburn, S. R., "Perfect hash families: Probabilistic methods and explicit constructions", Journal of Combinatorial Theory, Series A **92**, 1, 54–60 (2000).

[15] Blackburn, S. R., "Perfect hash families with few functions", Unpublished manuscript (2000).

[16] Blackburn, S. R., "Frameproof codes", SIAM Journal on Discrete Mathematics **16**, 3, 499–510 (2003).

[17] Blackburn, S. R., M. Burmester, Y. Desmedt and P. R. Wild, "Efficient multiplicative sharing schemes", in "Advances in Cryptology - EUROCRYPT '96", pp. 107–118 (1996).

[18] Blackburn, S. R. and P. R. Wild, "Optimal linear perfect hash families", Journal of Combinatorial Theory, Series A **83**, 2, 233–250 (1998).

[19] Brickell, E. F., "A problem in broadcast encryption", in "5th Vermont Summer Workshop on Combinatorics and Graph Theory", (1991).

[20] Brouwer, A. E., J. B. Shearer, N. J. A. Sloane and W. D. Smith, "A new table of constant weight codes", IEEE Transactions on Information Theory **36**, 6, 1334–1380 (1990).

[21] Cassels, J. and A. Godbole, "Covering arrays for some equivalence classes of words", Journal of Combinatorial Designs **27**, 8, 506–521 (2019).

[22] Cawse, J. N., *Experimental design for combinatorial and high throughput materials development* (GE Global Research Technical Report, 2002).

[23] Colbourn, C. J., "Combinatorial aspects of covering arrays", Le Matematiche (Catania) **58**, 121–167 (2004).

[24] Colbourn, C. J., "Covering array tables for t=2, 3, 4, 5, 6", URL `http://www.public.asu.edu/~ccolbou/src/tabby/catable.html` (2005–2018).

[25] Colbourn, C. J., "Constructing perfect hash families using a greedy algorithm", Coding and Cryptology pp. 109–118 (2008).

[26] Colbourn, C. J., "Distributing hash families and covering arrays", Journal of Combinatorics, Information, and System Sciences **34**, 113–126 (2009).

[27] Colbourn, C. J., "Covering arrays and hash families", NATO Science for Peace and Security Series, D: Information and Communication Security **29**, Information Security, Coding Theory and Related Combinatorics, 99–135 (2011).

[28] Colbourn, C. J. and J. H. Dinitz, *Handbook of Combinatorial Designs* (CRC Press, 2007).

[29] Colbourn, C. J. and R. E. Dougherty, "Fractal perfect hash families (extended abstract)", Electronic Notes in Discrete Mathematics **65**, 37–42, 7th International Conference on Algebraic Informatics (CAI 2017): Design Theory Track (2018).

[30] Colbourn, C. J., R. E. Dougherty and D. Horsley, "Distributing hash families with few rows", Theoretical Computer Science (Accepted) (2018).

[31] Colbourn, C. J., D. Horsley and V. R. Syrotiuk, "Strengthening hash families and compressive sensing", Journal of Discrete Algorithms **16**, 170–186 (2012).

[32] Colbourn, C. J., D. Jungnickel and A. Rosa, "The strong chromatic number of partial triple systems", Discrete Applied Mathematics **20**, 1, 31–38 (1988).

[33] Colbourn, C. J. and E. Lanus, "Subspace restrictions for covering perfect hash families", Art of Discrete and Applied Mathematics **1**, #P02.03 (2018).

[34] Colbourn, C. J., E. Lanus and K. Sarkar, "Asymptotic and constructive methods for covering perfect hash families and covering arrays", Designs, Codes and Cryptography **86**, 4, 907–937 (2017).

[35] Colbourn, C. J. and A. C. H. Ling, "Linear hash families and forbidden configurations", Designs, Codes and Cryptography **52**, 1, 25–55 (2009).

[36] Colbourn, C. J. and A. C. H. Ling, "A recursive construction for perfect hash families", Journal of Mathematical Cryptology **3**, 4, 291–306 (2009).

[37] Colbourn, C. J. and D. W. McClary, "Locating and detecting arrays for interaction faults", Journal of Combinatorial Optimization **15**, 1, 17–48 (2008).

[38] Colbourn, C. J. and J. Torres-Jimenez, "Heterogeneous hash families and covering arrays", Contemporary Mathematics **523**, 3–15 (2010).

[39] Czech, Z. J., G. Havas and B. S. Majewski, "Perfect hashing", Theoretical Computer Science **182**, 1–143 (1997).

[40] Damaschke, P., "Adaptive versus nonadaptive attribute-efficient learning", Machine Learning **41**, 2, 197–215 (2000).

[41] Davis, L., "Job shop scheduling with genetic algorithms", in "Proceedings of an international conference on genetic algorithms and their applications", pp. 136–140 (1985).

[42] Deb, K., A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", IEEE Transactions on Evolutionary Computation **6**, 2, 182–197 (2002).

[43] Deng, D., D. R. Stinson and R. Wei, "The lovász local lemma and its applications to some combinatorial arrays", Designs, Codes and Cryptography **32**, 1, 121–134 (2004).

[44] Dinitz, J. H., A. C. H. Ling and D. R. Stinson, "Perfect hash families from transversal designs", The Australasian Journal of Combinatorics **37**, 233–242 (2007).

[45] Dougherty, R. E., "Perfect hash family tables for t=3 to 11", URL `http://www.public.asu.edu/~redoughe/phf_pages/phf_tables.html` (2017).

[46] Dougherty, R. E. and C. J. Colbourn, "Perfect hash families: The generalization to higher indices", (Submitted) (2019).

[47] Dougherty, R. E., E. Lanus, C. J. Colbourn and S. Forrest, "Genetic algorithms for affine transformations to existential t-restrictions", in "Proceedings of the Genetic and Evolutionary Computation Conference Companion", GECCO '19, pp. 1707–1708 (ACM, New York, NY, USA, 2019).

[48] Eén, N. and A. Biere, "Effective preprocessing in sat through variable and clause elimination", in "International conference on theory and applications of satisfiability testing", pp. 61–75 (Springer, 2005).

[49] Fiat, A. and M. Naor, "Broadcast encryption", in "Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology", CRYPTO '93, pp. 480–491 (1994).

[50] Fredman, M. and J. Komlós, "On the Size of Separating Systems and Families of Perfect Hash Functions", SIAM Journal on Algebraic Discrete Methods **5**, 1, 61–68 (1984).

[51] Fuji-Hara, R., "Perfect hash families of strength three with three rows from varieties on finite projective geometries", Designs, Codes and Cryptography **77**, 2, 351–356 (2015).

[52] Hnich, B., S. D. Prestwich, E. Selensky and B. M. Smith, "Constraint models for the covering test problem", Constraints **11**, 2, 199–219 (2006).

[53] Knill, E., W. J. Bruno and D. C. Torney, "Non-adaptive group testing in the presence of errors", Discrete Applied Mathematics **88**, 1, 261–290 (1998).

[54] Kuhn, D. R. and M. J. Reilly, "An investigation of the applicability of design of experiments to software testing", in "27th Annual NASA Goddard/IEEE Software Engineering Workshop", pp. 91–95 (2002).

[55] Leach, K., R. Dougherty, C. Spensky, S. Forrest and W. Weimer, "Evolutionary computation for improving malware analysis", in "Proceedings of the 6th International Workshop on Genetic Improvement", GI '19, pp. 18–19 (IEEE Press, Piscataway, NJ, USA, 2019).

[56] MacWilliams, F. J. and N. J. A. Sloane, *The theory of error-correcting codes* (North-Holland Publishing Co., Amsterdam-New York-Oxford, 1977).

[57] Martirosyan, S. and T. van Trung, "Explicit constructions for perfect hash families", Designs, Codes and Cryptography **46**, 1, 97–112 (2008).

[58] Mehlhorn, K., "On the program size of perfect and universal hash functions", in "23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)", pp. 170–175 (1982).

[59] Moser, R. A. and G. Tardos, "A constructive proof of the general lovász local lemma", Journal of the ACM **57**, 2, 11:1–11:15 (2010).

[60] Moura, L., S. Raaphorst and B. Stevens, “The lovász local lemma and variable strength covering arrays”, Electronic Notes in Discrete Mathematics **65**, 43–49, 7th International Conference on Algebraic Informatics (CAI 2017): Design Theory Track (2018).

[61] Newman, I. and A. Wigderson, “Lower bounds on formula size of Boolean functions using hypergraph entropy”, SIAM Journal on Discrete Mathematics **8**, 4, 536–542 (1995).

[62] Niu, X. and H. Cao, “Constructions and bounds for separating hash families”, Discrete Mathematics **341**, 9, 2627–2638 (2018).

[63] Procacci, A. and R. Sanchis, “Perfect and separating hash families: new bounds via the algorithmic cluster expansion local lemma”, Annales de l’Institut Henri Poincaré D. Combinatorics, Physics and their Interactions **5**, 2, 153–171 (2018).

[64] Rodriguez-Tello, E. and J. Torres-Jimenez, “Memetic algorithms for constructing binary covering arrays of strength three”, in “International Conference on Artificial Evolution (Evolution Artificielle)”, pp. 86–97 (Springer, 2009).

[65] Sabharwal, S., P. Bansal and N. Mittal, “Construction of t-way covering arrays using genetic algorithm”, International Journal of System Assurance Engineering and Management **8**, 2, 264–274 (2017).

[66] Sarkar, K., *Covering Arrays: Algorithms and Asymptotics*, Ph.D. thesis, Arizona State University (2016).

[67] Sarkar, K. and C. J. Colbourn, “Upper bounds on the size of covering arrays”, SIAM Journal on Discrete Mathematics **31**, 2, 1277–1293 (2017).

[68] Seidel, S. A., K. Sarkar, C. J. Colbourn and V. R. Syrotiuk, “Separating interaction effects using locating and detecting arrays”, in “International Workshop on Combinatorial Algorithms”, pp. 349–360 (2018).

[69] Shangguan, C. and G. Ge, “Separating hash families: A johnson-type bound and new constructions”, SIAM Journal on Discrete Mathematics **30**, 4, 2243–2264 (2016).

[70] Shasha, D. E., A. Y. Kouranov, L. V. Lejay, M. F. Chou and G. M. Coruzzi, “Using combinatorial design to study regulation by multiple input signals. a tool for parsimony in the post-genomics era”, Plant Physiology **127**, 4, 1590–1594 (2001).

[71] Sherwood, G. B., S. S. Martirosyan and C. J. Colbourn, “Covering arrays of higher strength from permutation vectors”, Journal of Combinatorial Designs **14**, 3, 202–213 (2006).

[72] Staddon, J. N., D. R. Stinson and R. Wei, “Combinatorial properties of frameproof and traceability codes”, IEEE Transactions on Information Theory **47**, 1042–1049 (2001).

[73] Stardom, J., *Metaheuristics and the search for covering and packing arrays*, Master's thesis, Simon Frasier University (2001).

[74] Stinson, D. R., "On some methods for unconditionally secure key distribution and broadcast encryption", Designs, Codes and Cryptography **12**, 3, 215–243 (1997).

[75] Stinson, D. R., T. van Trung and R. Wei, "Secure frameproof codes, key distribution patterns, group testing algorithms and related structures", Journal of Statistical Planning and Inference **86**, 2, 595–617 (2000).

[76] Stinson, D. R., R. Wei and L. Zhu, "New constructions for perfect hash families and related structures using combinatorial designs and codes", Journal of Combinatorial Designs **8**, 3, 189–200 (2000).

[77] Timaná-Peña, J. A., C. A. Cobos-Lozada and J. Torres-Jimenez, "Metaheuristic algorithms for building covering arrays: A review", Facultad de Ingeniería **25**, 43, 31–45 (2016).

[78] Van Den Berg, E., E. Candès, G. Chinn, C. Levin, P. D. Olcott and C. Sing-Long, "Single-photon sampling architecture for solid-state imaging sensors", Proceedings of the National Academy of Sciences **110**, 30, E2752–E2761 (2013).

[79] Walker II, R. A. and C. J. Colbourn, "Perfect hash families: Constructions and existence", Journal of Mathematical Cryptology **1**, 2, 125–150 (2007).

APPENDIX A

EXTENDED TABLES FOR FRACTAL

Table A.1: Further Improvements for Strength 6, Four Rows

v	37	46	61	64	73	82	85	91	97	106	133	136
$k_{fractal}$	54	68	92	96	111	124	130	140	149	164	206	212
k_{old}	53	67	90	95	110	123	128	137	147	163	204	210
v	145	151	181	190	199	232	235	241	253	265	271	274
$k_{fractal}$	227	236	285	300	314	368	372	383	402	422	432	436
k_{old}	226	235	283	299	311	366	371	382	397	411	423	429
v	289	298	313	316	331	337	352	361				
$k_{fractal}$	461	476	500	504	530	539	564	579				
k_{old}	459	475	493	496	508	532	562	578				
v	379	385	391	397	406	409	421	430	451	460	463	
$k_{fractal}$	608	617	628	638	652	656	677	692	726	740	746	
k_{old}	599	609	615	627	645	651	675	691	715	726	732	
v	469	481	487	496	505	511	514	529	541	547	562	
$k_{fractal}$	756	775	784	800	815	824	828	854	874	884	908	
k_{old}	744	768	780	798	814	823	826	841	859	871	901	
v	568	571	577	586	595	601	613	625	628	631	649	
$k_{fractal}$	916	922	933	948	962	971	992	1012	1016	1022	1051	
k_{old}	913	919	931	947	959	965	977	996	1002	1008	1044	
v	661	664	673	685	691	694	703	715	721	727	742	
$k_{fractal}$	1070	1076	1091	1110	1118	1124	1140	1160	1170	1178	1204	
k_{old}	1068	1074	1090	1105	1111	1114	1123	1145	1156	1167	1197	
v	757	766	781	793	799	811	817	820	826	829	841	
$k_{fractal}$	1229	1244	1268	1287	1298	1318	1328	1332	1340	1346	1367	
k_{old}	1227	1243	1261	1273	1279	1300	1312	1318	1330	1336	1360	
v	856	859	865	883	898	901	913	919	925	937	946	961
$k_{fractal}$	1392	1396	1407	1436	1460	1466	1486	1496	1505	1524	1540	1565
k_{old}	1390	1395	1406	1427	1442	1445	1467	1479	1491	1515	1533	1563
v	970	991	1009	1021	1027	1036	1051	1057	1072	1081	1084	1093
$k_{fractal}$	1580	1614	1644	1664	1674	1688	1712	1723	1748	1763	1768	1781
k_{old}	1579	1603	1628	1645	1656	1674	1704	1716	1746	1762	1768	1777
v	1105	1123	1126	1135	1141	1153	1171	1174	1177	1189	1198	1216
$k_{fractal}$	1802	1832	1836	1852	1862	1881	1910	1916	1919	1941	1956	1984
k_{old}	1789	1807	1813	1831	1843	1867	1903	1909	1915	1939	1955	1976

Table A.2: Further Improvements for Strength 6, Four Rows, Part 2

v	1225	1243	1249	1255	1261	1276	1297	1312	1321	1327	1345	1351
$k_{fractal}$	2000	2030	2039	2050	2060	2084	2119	2144	2159	2168	2197	2208
k_{old}	1985	2004	2016	2028	2040	2070	2112	2142	2158	2167	2185	2191
v	1369	1378	1381	1387	1393	1396	1405	1426	1429	1441	1450	1459
$k_{fractal}$	2238	2252	2258	2268	2276	2280	2297	2332	2336	2357	2372	2386
k_{old}	2211	2229	2235	2247	2259	2265	2283	2325	2331	2355	2371	2383
v	1480	1483	1489	1501	1513	1519	1531	1540	1561	1567	1576	1585
$k_{fractal}$	2420	2426	2434	2456	2476	2486	2504	2520	2555	2564	2580	2595
k_{old}	2404	2407	2413	2428	2452	2464	2488	2506	2548	2560	2578	2594
v	1597	1621	1639	1651	1654	1657	1675	1681	1684	1702	1711	1717
$k_{fractal}$	2614	2654	2684	2704	2708	2714	2742	2753	2756	2788	2802	2813
k_{old}	2609	2633	2655	2679	2685	2691	2727	2739	2745	2781	2799	2811
v	1726	1741	1765	1768	1783	1795	1801	1825	1828	1837	1849	1861
$k_{fractal}$	2828	2852	2892	2896	2922	2942	2952	2990	2996	3009	3031	3050
k_{old}	2827	2845	2869	2872	2892	2916	2928	2976	2982	3000	3024	3048
v	1864	1873	1882	1891	1915	1921	1933	1945	1951	1954	1981	2002
$k_{fractal}$	3056	3071	3084	3100	3140	3149	3170	3190	3200	3204	3249	3284
k_{old}	3054	3070	3082	3091	3115	3124	3147	3169	3175	3181	3235	3277
v	2017	2026	2041	2047	2053	2071	2080	2089	2101	2107	2113	2140
$k_{fractal}$	3309	3324	3347	3358	3366	3398	3412	3428	3448	3458	3467	3512
k_{old}	3307	3323	3341	3347	3353	3371	3380	3396	3420	3432	3444	3498
v	2143	2161	2176	2179	2185	2206	2209	2221	2233	2245	2251	2263
$k_{fractal}$	3516	3547	3572	3576	3587	3620	3626	3644	3666	3685	3696	3716
k_{old}	3504	3540	3570	3575	3586	3610	3613	3625	3637	3651	3663	3687
v	2269	2278	2281	2305	2311	2326	2332	2341	2350	2377	2401	2416
$k_{fractal}$	3726	3740	3743	3785	3794	3820	3828	3845	3860	3904	3944	3968
k_{old}	3699	3717	3723	3771	3783	3813	3825	3843	3859	3889	3921	3936
v	2419	2431	2437	2443	2449	2458	2461	2476	2485	2497		
$k_{fractal}$	3974	3994	4004	4012	4023	4036	4040	4068	4082	4103		
k_{old}	3940	3964	3976	3988	4000	4018	4024	4054	4072	4096		

Table A.3: Further Improvements for Strength 6, Five Rows

v	36	42	254	263	273	281	292	306	320	329	336	344
$k_{fractal}$	60	70	590	615	645	665	700	750	800	825	840	860
k_{old}	57	67	502	511	521	529	580	594	613	653	660	668
v	352	360	370	377	384	394	400	409	416	425	433	441
$k_{fractal}$	880	900	930	945	960	990	1000	1025	1040	1065	1085	1105
k_{old}	676	684	729	800	816	826	832	841	848	857	865	873
v	452	457	466	473	484	491	501	510	522	630	639	648
$k_{fractal}$	1140	1145	1170	1185	1220	1235	1265	1290	1330	1750	1775	1800
k_{old}	884	889	898	905	964	971	997	1260	1290	1398	1407	1416
v	675	685	693	703	711	720	729	738	747	756	765	774
$k_{fractal}$	1875	1905	1925	1955	1975	2000	2025	2050	2075	2100	2125	2150
k_{old}	1443	1453	1461	1471	1479	1488	1497	1506	1530	1620	1629	1638
v	784	792	802	811	819	830	838	848	860	865	878	887
$k_{fractal}$	2180	2200	2230	2255	2275	2310	2330	2360	2400	2405	2450	2475
k_{old}	1648	1656	1666	1675	1683	1694	1702	1712	1724	1729	1746	1755

Table A.4: Further Improvements for Strength 7, Five Rows

v	40	52	53	56	57	59	61	62	65	66	69	70
$k_{fractal}$	56	74	76	80	81	85	88	89	94	97	100	101
k_{old}	55	73	74	79	80	83	86	87	93	95	99	100
v	71	72	73	74	75	76	77	83	86	90	91	92
$k_{fractal}$	103	104	105	106	109	112	113	123	128	134	135	137
k_{old}	101	102	103	105	106	108	110	119	126	132	133	135
v	93	95	96	99	100	101	102	103	104	105	106	107
$k_{fractal}$	138	143	144	148	149	150	154	155	156	157	158	160
k_{old}	137	141	143	147	148	149	150	151	153	155	157	159
v	108	109	110	111	112	116	117	118	120	121	122	125
$k_{fractal}$	162	167	168	169	170	176	177	178	180	182	184	191
k_{old}	161	163	164	166	167	172	173	174	177	179	181	187
v	126	251	252	253	255	256	257	259	260	262	263	265
$k_{fractal}$	192	400	401	402	404	408	409	414	417	420	421	426
k_{old}	188	389	391	393	397	399	401	403	404	406	407	409
v	266	267	270	272	273	274	275	276	279	280	281	283
$k_{fractal}$	428	429	432	434	435	436	439	444	447	448	452	455
k_{old}	410	411	414	416	417	418	419	420	423	424	425	427
v	284	286	287	291	292	294	296	297	298	299	301	303
$k_{fractal}$	458	462	464	469	472	474	476	477	478	480	484	490
k_{old}	428	430	431	435	436	438	440	441	442	443	457	447
v	305	308	309	311	312	313	315	318	319	321	322	324
$k_{fractal}$	492	495	500	503	504	508	510	513	518	520	522	524
k_{old}	453	464	465	467	468	481	491	498	499	501	502	504

Table A.5: Further Improvements for Strength 7, Five Rows, Part 2

v	326	327	328	331	333	334	335	336	337	338	340	341
$k_{fractal}$	528	529	530	534	537	538	539	542	547	550	552	553
k_{old}	506	507	508	511	513	514	515	516	517	518	520	521
v	342	343	345	346	348	351	352	354	355	356	357	358
$k_{fractal}$	554	559	561	562	568	571	572	574	575	576	577	578
k_{old}	522	523	525	526	528	531	532	534	535	536	537	538
v	359	361	363	365	366	370	372	375	376	379	384	386
$k_{fractal}$	579	589	592	594	598	602	609	614	616	620	626	628
k_{old}	539	541	543	571	576	572	582	585	586	589	594	596
v	389	390	391	393	394	395	396	397	398	400	404	405
$k_{fractal}$	634	636	637	639	640	641	642	644	648	656	660	662
k_{old}	599	600	601	603	604	605	606	609	614	620	624	625
v	406	408	410	411	412	414	415	416	417	418	420	421
$k_{fractal}$	664	670	672	675	676	678	679	680	681	682	684	688
k_{old}	626	628	630	631	632	634	635	640	641	642	644	645
v	423	425	427	431	432	434	435	438	441	442	445	448
$k_{fractal}$	690	696	702	706	708	710	714	721	728	729	732	736
k_{old}	647	651	655	663	665	669	671	677	683	685	691	697
v	449	452	453	455	457	458	460	461	462	464	465	466
$k_{fractal}$	737	740	741	746	751	752	756	758	759	761	765	770
k_{old}	699	705	707	711	721	722	724	725	726	729	731	733
v	470	471	473	474	475	476	478	479	481	482	483	484
$k_{fractal}$	774	777	779	780	783	786	788	790	793	795	796	798
k_{old}	741	743	747	749	751	753	757	759	763	765	767	769
v	488	490	491	492	493	494	495	496	497	498	499	502
$k_{fractal}$	802	804	805	807	808	810	812	816	817	820	823	826
k_{old}	776	780	782	784	786	787	789	791	793	795	797	802
v	503	504	506	507	508	512	515	516	518	519	522	523
$k_{fractal}$	828	831	839	840	844	848	851	852	854	856	859	860
k_{old}	803	804	806	807	808	812	815	816	820	825	828	829
v	525	529	531	532	533	534	535	536	538	541	544	545
$k_{fractal}$	870	874	876	878	879	880	881	884	890	897	900	902
k_{old}	831	835	837	838	839	840	841	842	844	847	850	851
v	546	550	552	553	554	555	557	558	559	561	564	565
$k_{fractal}$	904	910	914	917	918	921	924	925	926	929	932	937
k_{old}	852	856	858	859	860	863	869	870	871	873	876	877
v	569	570	571	573	574	577	581	582	584	585	586	588
$k_{fractal}$	942	944	949	952	953	961	965	966	968	972	974	979
k_{old}	881	882	883	885	886	889	893	894	896	897	898	900
v	592	593	597	598	599	601	602	604	606	607	610	611
$k_{fractal}$	983	984	988	989	991	995	996	1002	1008	1009	1012	1015
k_{old}	904	905	909	910	933	943	944	946	948	949	952	953

Table A.6: Further Improvements for Strength 7, Five Rows, Part 3

v	612	613	615	616	618	620	622	623	624	626	628	630
$k_{fractal}$	1016	1017	1019	1022	1026	1028	1034	1035	1036	1042	1044	1050
k_{old}	954	955	957	958	960	962	978	983	988	990	992	994
v	633	635	639	640	644	645	647	648	651	653	655	656
$k_{fractal}$	1053	1060	1064	1066	1070	1072	1079	1080	1084	1088	1091	1092
k_{old}	997	999	1003	1004	1008	1009	1011	1012	1015	1017	1019	1020
v	658	659	661	663	665	667	668	671	672	675	678	680
$k_{fractal}$	1096	1100	1102	1106	1110	1112	1116	1120	1121	1124	1128	1130
k_{old}	1022	1023	1025	1027	1029	1031	1032	1035	1036	1039	1042	1044
v	682	683	686	687	690	691	692	693	694	695	696	697
$k_{fractal}$	1135	1139	1142	1143	1151	1153	1154	1155	1156	1160	1161	1162
k_{old}	1058	1063	1050	1059	1070	1071	1072	1101	1106	1111	1116	1117
v	698	699	702	704	707	709	710	712	713	716	717	719
$k_{fractal}$	1165	1166	1172	1176	1184	1189	1190	1192	1193	1196	1197	1201
k_{old}	1118	1119	1122	1124	1127	1129	1130	1132	1133	1136	1137	1139
v	721	722	723	726	727	728	729	730	732	733	734	735
$k_{fractal}$	1204	1205	1208	1211	1214	1215	1216	1217	1222	1223	1226	1230
k_{old}	1141	1142	1143	1146	1147	1148	1149	1150	1152	1153	1154	1155
v	737	740	744	749	753	754	755	759	760	761	762	765
$k_{fractal}$	1232	1240	1244	1256	1260	1261	1265	1270	1272	1273	1274	1282
k_{old}	1157	1160	1165	1175	1183	1185	1187	1195	1197	1199	1201	1207
v	767	769	770	772	776	778	783	784	785	786	788	790
$k_{fractal}$	1284	1286	1287	1296	1300	1304	1309	1310	1313	1314	1324	1326
k_{old}	1211	1215	1217	1221	1229	1240	1243	1246	1247	1249	1253	1257
v	793	795	796	802	804	809	811	814	816	817	818	819
$k_{fractal}$	1332	1335	1338	1346	1348	1358	1362	1366	1368	1370	1371	1372
k_{old}	1263	1267	1269	1281	1285	1295	1299	1305	1309	1311	1313	1315
v	821	826	827	828	829	830	831	832	833	834	835	838
$k_{fractal}$	1380	1385	1387	1388	1389	1390	1395	1396	1398	1402	1403	1410
k_{old}	1319	1329	1331	1333	1335	1337	1339	1341	1343	1345	1347	1353
v	839	841	845	847	851	852	854	855	856	857	858	862
$k_{fractal}$	1412	1416	1422	1424	1428	1432	1434	1435	1436	1437	1440	1449
k_{old}	1355	1359	1367	1371	1378	1380	1384	1386	1388	1390	1391	1399
v	863	864	867	871	873	876	877	880	882	884	886	890
$k_{fractal}$	1450	1452	1455	1465	1467	1470	1471	1484	1486	1492	1494	1500
k_{old}	1401	1403	1407	1411	1413	1416	1417	1420	1422	1424	1426	1430
v	892	893	895	896	897	898	900	903	904	907	908	909
$k_{fractal}$	1504	1505	1507	1508	1509	1510	1512	1520	1522	1527	1528	1532
k_{old}	1432	1433	1435	1436	1437	1438	1444	1447	1448	1451	1452	1453
v	912	913	915	918	920	921	927	931	934	935	941	943
$k_{fractal}$	1535	1536	1538	1548	1555	1556	1564	1568	1574	1578	1584	1586
k_{old}	1456	1457	1459	1462	1464	1465	1471	1475	1478	1479	1485	1488
v	944	947	949	950	951	952	953	954	956	958	961	963
$k_{fractal}$	1591	1594	1596	1597	1598	1602	1605	1606	1608	1612	1616	1624
k_{old}	1490	1496	1500	1502	1504	1506	1508	1510	1514	1518	1524	1528

Table A.7: Further Improvements for Strength 7, Five Rows, Part 4

v	964	966	967	968	969	970	971	972	974	977	979	982
$k_{fractal}$	1626	1628	1630	1633	1635	1636	1637	1638	1640	1652	1654	1658
k_{old}	1530	1534	1536	1538	1540	1542	1544	1546	1550	1556	1559	1565
v	983	984	985	986	987	990	995	998	999	1002	1003	1005
$k_{fractal}$	1659	1664	1668	1670	1671	1674	1679	1682	1685	1688	1689	1692
k_{old}	1567	1580	1585	1572	1574	1586	1595	1598	1599	1602	1603	1605
v	1006	1009	1011	1012	1013	1014	1015	1016	1017	1018	1019	1021
$k_{fractal}$	1694	1702	1704	1708	1709	1711	1712	1713	1714	1718	1719	1727
k_{old}	1606	1609	1611	1612	1613	1614	1615	1616	1617	1618	1619	1621
v	1023	1024	1026	1027	1029	1030	1032	1033	1034	1035	1036	1038
$k_{fractal}$	1729	1732	1734	1736	1738	1740	1742	1746	1750	1752	1756	1758
k_{old}	1623	1624	1626	1627	1629	1630	1632	1633	1638	1643	1648	1650
v	1039	1040	1041	1042	1044	1045	1046	1047	1048	1051	1056	1060
$k_{fractal}$	1759	1760	1761	1762	1764	1765	1766	1768	1769	1777	1784	1788
k_{old}	1651	1652	1653	1654	1656	1657	1658	1659	1660	1663	1668	1672
v	1061	1062	1063	1064	1065	1068	1071	1072	1074	1077	1078	1082
$k_{fractal}$	1789	1790	1791	1792	1797	1808	1812	1814	1816	1822	1824	1830
k_{old}	1673	1674	1675	1680	1685	1692	1695	1696	1698	1701	1702	1706
v	1084	1087	1088	1089	1091	1092	1095	1096	1098	1099	1100	1104
$k_{fractal}$	1832	1839	1840	1845	1847	1852	1855	1856	1858	1859	1862	1866
k_{old}	1708	1711	1712	1713	1715	1716	1719	1742	1722	1723	1724	1754
v	1106	1107	1108	1110	1111	1115	1116	1117	1119	1121	1124	1125
$k_{fractal}$	1868	1871	1872	1878	1880	1890	1891	1892	1894	1896	1906	1908
k_{old}	1756	1757	1758	1760	1761	1765	1766	1767	1769	1771	1774	1797
v	1131	1133	1134	1135	1136	1137	1138	1139	1141	1145	1146	1149
$k_{fractal}$	1916	1918	1920	1922	1926	1927	1930	1934	1940	1944	1945	1948
k_{old}	1815	1817	1818	1819	1820	1821	1822	1823	1825	1829	1830	1833
v	1150	1155	1157	1158	1159	1162	1166	1167	1168	1169	1171	1172
$k_{fractal}$	1950	1955	1961	1964	1965	1970	1974	1975	1976	1979	1982	1984
k_{old}	1834	1839	1841	1842	1843	1846	1850	1851	1852	1853	1855	1856
v	1175	1178	1179	1184	1185	1186	1187	1190	1191	1195	1196	1204
$k_{fractal}$	1988	2000	2001	2006	2007	2009	2012	2022	2023	2032	2036	2044
k_{old}	1861	1867	1869	1879	1881	1883	1885	1891	1893	1901	1903	1919
v	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1217	1218
$k_{fractal}$	2046	2047	2048	2049	2050	2051	2052	2053	2055	2056	2058	2059
k_{old}	1923	1925	1927	1929	1931	1933	1935	1937	1939	1941	1945	1947
v	1219	1221	1225	1227	1228	1231	1233	1234	1237	1238	1240	1241
$k_{fractal}$	2064	2071	2077	2084	2085	2088	2091	2092	2097	2102	2106	2110
k_{old}	1949	1968	1980	1982	1984	1987	1989	1990	1993	1994	1996	1997
v	1243	1244	1245	1246	1250	1253	1254	1255	1258	1259	1263	1265
$k_{fractal}$	2112	2113	2116	2120	2127	2131	2134	2136	2139	2140	2144	2148
k_{old}	1999	2000	2001	2003	2011	2017	2019	2021	2027	2029	2037	2041

Table A.8: Further Improvements for Strength 7, Five Rows, Part 5

v	1268	1269	1270	1273	1276	1279	1280	1285	1286	1291	1292	1295
$k_{fractal}$	2151	2152	2154	2163	2168	2171	2172	2185	2186	2193	2198	2201
k_{old}	2047	2049	2051	2057	2063	2069	2071	2081	2083	2093	2095	2101
v	1296	1297	1299	1300	1304	1306	1308	1310	1314	1316	1317	1319
$k_{fractal}$	2202	2203	2211	2212	2220	2222	2228	2230	2234	2240	2241	2243
k_{old}	2103	2105	2109	2111	2119	2123	2127	2131	2139	2143	2145	2149
v	1320	1321	1323	1325	1330	1332	1333	1335	1336	1337	1339	1340
$k_{fractal}$	2244	2248	2250	2252	2257	2259	2264	2266	2268	2273	2276	2277
k_{old}	2151	2153	2157	2161	2171	2175	2177	2181	2183	2185	2189	2191
v	1342	1345	1346	1347	1348	1351	1357	1358	1363	1364	1368	1369
$k_{fractal}$	2284	2288	2289	2292	2293	2305	2312	2313	2318	2321	2328	2332
k_{old}	2195	2201	2203	2205	2207	2213	2225	2226	2236	2238	2245	2247
v	1370	1371	1373	1374	1377	1378	1379	1382	1383	1386	1388	1390
$k_{fractal}$	2333	2337	2340	2341	2344	2348	2349	2352	2355	2358	2362	2370
k_{old}	2249	2251	2255	2256	2259	2260	2261	2264	2265	2268	2270	2272
v	1391	1395	1398	1399	1401	1404	1405	1408	1410	1412	1414	1416
$k_{fractal}$	2371	2375	2378	2379	2384	2392	2393	2396	2400	2403	2408	2410
k_{old}	2273	2277	2280	2281	2283	2286	2287	2290	2292	2294	2296	2298
v	1417	1418	1419	1421	1423	1424	1425	1426	1427	1429	1430	1432
$k_{fractal}$	2413	2415	2416	2422	2424	2426	2427	2429	2430	2433	2434	2438
k_{old}	2299	2300	2301	2303	2305	2306	2307	2308	2309	2311	2312	2314
v	1434	1436	1440	1442	1444	1446	1447	1451	1452	1453	1455	1458
$k_{fractal}$	2442	2446	2452	2454	2460	2462	2464	2468	2469	2470	2472	2486
k_{old}	2316	2318	2322	2324	2326	2328	2329	2333	2334	2335	2337	2340
v	1460	1461	1465	1467	1468	1470	1474	1475	1476	1477	1480	1481
$k_{fractal}$	2493	2494	2498	2501	2502	2506	2512	2515	2516	2517	2524	2526
k_{old}	2342	2343	2347	2349	2350	2352	2356	2357	2358	2359	2362	2363
v	1483	1486	1489	1490	1491	1492	1495	1496	1497	1499	1501	1502
$k_{fractal}$	2530	2534	2542	2543	2545	2548	2551	2552	2554	2556	2558	2559
k_{old}	2367	2398	2413	2414	2415	2416	2419	2420	2421	2423	2425	2426
v	1503	1505	1507	1509	1515	1519	1520	1525	1527	1531	1532	1533
$k_{fractal}$	2560	2568	2574	2580	2586	2590	2592	2604	2606	2610	2612	2616
k_{old}	2427	2429	2431	2433	2439	2443	2444	2449	2451	2455	2456	2457
v	1538	1540	1541	1542	1544	1545	1548	1549	1551	1552	1554	1556
$k_{fractal}$	2624	2626	2627	2630	2634	2637	2640	2641	2644	2648	2650	2656
k_{old}	2462	2464	2465	2468	2468	2469	2472	2473	2475	2476	2478	2480
v	1558	1560	1561	1564	1569	1575	1577	1579	1581	1586	1588	1589
$k_{fractal}$	2662	2664	2665	2672	2678	2686	2688	2690	2692	2712	2714	2716
k_{old}	2482	2484	2485	2488	2493	2504	2508	2512	2516	2526	2530	2532

Table A.9: Further Improvements for Strength 7, Five Rows, Part 6

v	1590	1591	1592	1594	1596	1604	1608	1609	1610	1611	1613	1617
$k_{fractal}$	2717	2718	2720	2722	2730	2740	2745	2748	2749	2753	2760	2768
k_{old}	2534	2536	2538	2542	2546	2562	2570	2572	2574	2595	2605	2609
v	1619	1629	1631	1633	1638	1639	1640	1649	1650	1651	1652	1655
$k_{fractal}$	2770	2780	2786	2788	2802	2806	2808	2817	2818	2819	2822	2828
k_{old}	2611	2621	2623	2625	2630	2632	2634	2652	2654	2656	2658	2664
v	1657	1664	1666	1668	1669	1670	1674	1676	1683	1685	1687	1688
$k_{fractal}$	2830	2846	2848	2856	2861	2862	2868	2870	2878	2880	2884	2888
k_{old}	2668	2682	2686	2690	2692	2694	2702	2706	2720	2724	2728	2730
v	1696	1698	1700	1702	1708	1709	1712	1716	1719	1722	1726	1727
$k_{fractal}$	2896	2898	2908	2910	2916	2918	2926	2930	2938	2944	2948	2950
k_{old}	2746	2750	2754	2758	2769	2771	2777	2784	2790	2795	2803	2805
v	1728	1730	1731	1732	1733	1735	1738	1739	1747	1748	1750	1751
$k_{fractal}$	2952	2955	2959	2960	2961	2964	2972	2974	2992	2996	2998	3000
k_{old}	2807	2810	2811	2812	2813	2815	2818	2819	2827	2828	2830	2831
v	1753	1759	1760	1762	1764	1766	1767	1771	1780	1782	1783	1788
$k_{fractal}$	3002	3008	3010	3012	3014	3016	3018	3037	3048	3050	3051	3064
k_{old}	2833	2839	2840	2842	2844	2846	2847	2851	2884	2886	2887	2892
v	1789	1795	1798	1800	1805	1806	1807	1812	1814	1816	1822	1823
$k_{fractal}$	3065	3075	3080	3082	3092	3093	3097	3104	3110	3112	3120	3122
k_{old}	2893	2899	2902	2904	2909	2910	2911	2921	2925	2929	2941	2943
v	1828	1832	1837	1844	1852	1856	1859	1861	1863	1865	1873	1877
$k_{fractal}$	3128	3132	3148	3164	3172	3176	3181	3188	3190	3200	3208	3216
k_{old}	2953	2961	2971	2985	3001	3009	3015	3019	3023	3027	3043	3051
v	1880	1883	1885	1886	1887	1890	1898	1906	1908	1916	1918	1927
$k_{fractal}$	3224	3228	3232	3236	3238	3242	3250	3258	3268	3282	3290	3304
k_{old}	3057	3063	3067	3069	3071	3077	3093	3109	3113	3129	3133	3151
v	1932	1938	1940	1942	1950	1956	1958	1962	1966	1968	1970	1971
$k_{fractal}$	3314	3320	3322	3328	3338	3344	3346	3354	3362	3368	3370	3372
k_{old}	3161	3173	3177	3181	3197	3209	3213	3221	3229	3233	3237	3239
v	1973	1974	1979	1991	1995	1997	2000					
$k_{fractal}$	3374	3378	3396	3408	3412	3414	3424					
k_{old}	3243	3245	3255	3279	3287	3291	3297					

Table A.10: Further Improvements for Strength 8, Five Rows

v	55	69	93	97	112	125	131	141	150	165	188	189
$k_{fractal}$	72	91	124	129	150	167	176	190	202	223	254	256
k_{old}	71	90	123	128	149	166	174	188	200	222	251	252
v	199	207	213	228	237	245	257	261	267	285	286	301
$k_{fractal}$	270	280	289	310	322	332	350	355	364	388	390	411
k_{old}	267	279	287	309	321	329	345	351	360	387	388	410
v	306	309	315	335	344	345	357	369	373	384	389	402
$k_{fractal}$	418	421	430	458	470	472	487	505	510	526	533	550
k_{old}	418	421	427	453	466	468	486	503	509	525	533	546
v	403	423	433	437	462	471	477	482	489	501	505	521
$k_{fractal}$	552	580	594	599	634	646	655	662	670	688	693	716
k_{old}	547	574	589	595	632	646	654	662	669	681	686	710
v	531	539	540	547	565	573	579	580	585	609	618	629
$k_{fractal}$	730	740	742	750	777	787	796	798	805	838	850	866
k_{old}	725	737	738	749	775	787	796	797	805	832	843	859
v	639	653	657	678	693	697	698	709	727	735	741	747
$k_{fractal}$	880	899	904	934	955	960	962	975	1002	1012	1021	1030
k_{old}	874	895	901	932	954	960	962	973	993	1005	1014	1023
v	757	776	785	801	816	821	825	829	855	874	875	885
$k_{fractal}$	1044	1070	1082	1105	1126	1133	1138	1143	1180	1206	1208	1222
k_{old}	1038	1066	1080	1103	1125	1133	1137	1141	1170	1199	1200	1215
v	891	909	917	923	934	949	954	963	972	993	1013	1017
$k_{fractal}$	1228	1255	1265	1274	1290	1311	1318	1330	1342	1372	1400	1405
k_{old}	1224	1251	1263	1272	1288	1310	1318	1327	1336	1362	1392	1398
v	1023	1052	1070	1071	1077	1092	1097	1111	1119	1125	1141	1161
$k_{fractal}$	1414	1454	1478	1480	1489	1510	1517	1536	1546	1555	1578	1606
k_{old}	1407	1450	1477	1479	1487	1509	1517	1531	1539	1545	1568	1597
v	1170	1171	1179	1205	1229	1230	1233	1245	1250	1269	1287	1288
$k_{fractal}$	1618	1620	1630	1667	1700	1702	1705	1723	1730	1756	1780	1782
k_{old}	1611	1612	1624	1663	1699	1700	1705	1722	1730	1749	1769	1770
v	1299	1319	1329	1333	1341	1347	1368	1393	1397	1408	1413	1437
$k_{fractal}$	1798	1826	1840	1845	1855	1864	1894	1929	1934	1950	1957	1990
k_{old}	1787	1817	1832	1838	1850	1859	1890	1927	1933	1949	1957	1981
v	1449	1461	1467	1487	1497	1506	1524	1525	1541	1566	1575	1581
$k_{fractal}$	2005	2023	2032	2060	2074	2086	2110	2112	2135	2170	2182	2191
k_{old}	1993	2011	2020	2050	2065	2079	2106	2107	2131	2168	2182	2190
v	1586	1589	1615	1644	1645	1653	1665	1675	1689	1701	1713	1717
$k_{fractal}$	2198	2201	2238	2278	2280	2290	2308	2322	2341	2356	2374	2379
k_{old}	2198	2201	2227	2268	2269	2280	2298	2313	2334	2352	2370	2376

Table A.11: Further Improvements for Strength 8, Five Rows, Part 2

v	1724	1749	1763	1764	1769	1781	1782	1803	1833	1837	1853	1863
$k_{fractal}$	2390	2425	2444	2446	2453	2468	2470	2500	2542	2547	2570	2584
k_{old}	2386	2423	2444	2445	2453	2465	2466	2487	2529	2535	2559	2574
v	1882	1911	1917	1920	1942	1957	1961	1962	1973	1985	1989	2001
$k_{fractal}$	2610	2650	2659	2662	2694	2715	2720	2722	2735	2753	2758	2776
k_{old}	2603	2646	2655	2660	2692	2714	2720	2722	2733	2745	2749	2761
v	2031	2040	2051	2061	2085	2119	2120	2127	2133	2145	2160	2165
$k_{fractal}$	2818	2830	2846	2860	2893	2940	2942	2950	2959	2977	2998	3005
k_{old}	2805	2818	2835	2850	2886	2937	2938	2949	2958	2975	2997	3005
v	2169	2198	2209	2239	2253	2259	2269	2277	2281	2298	2333	2337
$k_{fractal}$	3010	3050	3066	3108	3127	3136	3150	3160	3165	3190	3239	3244
k_{old}	3009	3038	3049	3094	3115	3124	3139	3151	3157	3183	3235	3241
v	2358	2373	2378	2387	2421	2427	2435	2457	2476	2477	2487	
$k_{fractal}$	3274	3295	3302	3314	3361	3370	3380	3412	3438	3440	3454	
k_{old}	3272	3294	3302	3311	3345	3353	3365	3398	3426	3428	3443	

Table A.12: Further Improvements for Strength 8, Six Rows

v	167	270	280	288	333	352	365	386	396	408	434	450
$k_{fractal}$	252	432	450	480	540	576	600	630	648	672	720	750
k_{old}	248	413	424	432	503	532	545	601	615	628	658	675
v	464	485	492	503	516	525	572	598	620	636	680	693
$k_{fractal}$	768	792	810	840	864	900	960	1008	1050	1080	1152	1176
k_{old}	716	749	756	767	788	825	884	910	969	994	1044	1057
v	705	737	756	779	788	790	810	926	939	960	971	996
$k_{fractal}$	1200	1260	1296	1320	1344	1350	1458	1584	1620	1650	1680	1728
k_{old}	1069	1157	1176	1219	1228	1230	1254	1466	1479	1504	1515	1540
v	1015	1040	1073	1088	1104	1116	1141	1154	1176	1205	1236	1293
$k_{fractal}$	1764	1800	1848	1890	1920	1944	1980	2016	2058	2100	2160	2268
k_{old}	1602	1640	1692	1712	1728	1740	1795	1838	1860	1899	1961	2061
v	1312	1323	1370	1386	1432	1470	1498	1520	1556	1571	1596	1703
$k_{fractal}$	2304	2352	2400	2430	2520	2592	2646	2688	2730	2772	2808	3024
k_{old}	2080	2091	2138	2154	2302	2352	2380	2433	2480	2495	2536	2715
v	1728	1769	1786	1820	1824	1849	1865	1886	1908	1920	1981	2001
$k_{fractal}$	3072	3120	3168	3234	3240	3276	3300	3360	3402	3456	3528	3564
k_{old}	2740	2825	2842	2924	2928	2953	2969	2990	3012	3024	3181	3201
v	2020	2046	2069	2102	2113	2144	2169	2216	2252	2303	2318	2340
$k_{fractal}$	3600	3630	3696	3744	3780	3840	3888	3960	4032	4116	4158	4200
k_{old}	3235	3270	3293	3434	3445	3476	3513	3560	3596	3707	3722	3744
v	2352	2402	2430	2464								
$k_{fractal}$	4224	4320	4374	4410								
k_{old}	3756	3806	3881	3976								

Table A.13: Further Improvements for Strength 9, Six Rows

v	55	68	89	94	106	120	123	131	140	152	172	206
$k_{fractal}$	72	90	120	126	144	162	168	180	192	210	234	288
k_{old}	71	89	118	125	141	161	166	176	188	207	232	285
v	215	256	268	272	278	281	297	305	326	331	338	342
$k_{fractal}$	300	360	378	384	390	396	420	432	462	468	480	486
k_{old}	299	343	375	384	390	393	409	420	446	465	477	486
v	355	371	379	384	404	413	416	420	437	442	453	461
$k_{fractal}$	504	528	540	546	576	588	594	600	624	630	648	660
k_{old}	499	515	523	528	552	588	591	600	617	622	633	641
v	470	481	490	502	506	527	536	543	551	564	569	584
$k_{fractal}$	672	684	702	720	726	756	768	780	792	810	816	840
k_{old}	650	661	670	717	726	747	756	763	775	788	793	808
v	596	600	616	625	638	641	649	666	675	686	698	702
$k_{fractal}$	858	864	882	900	918	924	936	960	972	990	1008	1014
k_{old}	855	864	880	889	902	905	913	934	953	974	1005	1014
v	707	712	731	747	755	767	776	786	789	796	808	812
$k_{fractal}$	1020	1026	1056	1080	1092	1104	1122	1134	1140	1152	1170	1176
k_{old}	1019	1024	1043	1059	1067	1079	1088	1098	1101	1117	1167	1176
v	821	830	845	860	861	866	869	894	911	914	926	930
$k_{fractal}$	1188	1200	1224	1242	1248	1254	1260	1296	1320	1326	1344	1350
k_{old}	1185	1194	1209	1224	1225	1230	1233	1258	1275	1278	1341	1350
v	943	953	956	967	983	991	1001	1020	1040	1052	1056	1073
$k_{fractal}$	1368	1380	1386	1404	1428	1440	1452	1482	1512	1530	1536	1560
k_{old}	1363	1373	1376	1387	1403	1416	1431	1460	1480	1527	1536	1553
v	1090	1097	1113	1121	1126	1136	1139	1154	1174	1179	1186	1190
$k_{fractal}$	1584	1596	1620	1632	1638	1650	1656	1680	1710	1716	1728	1734
k_{old}	1570	1577	1593	1601	1606	1616	1619	1634	1682	1713	1725	1734
v	1211	1226	1232	1235	1251	1259	1268	1285	1296	1316	1325	1328
$k_{fractal}$	1764	1782	1794	1800	1824	1836	1848	1872	1890	1920	1932	1938
k_{old}	1755	1770	1776	1779	1795	1803	1812	1829	1840	1860	1932	1935
v	1332	1338	1357	1381	1397	1405	1418	1439	1444	1446	1466	1478
$k_{fractal}$	1944	1950	1980	2016	2040	2052	2070	2100	2106	2112	2142	2160
k_{old}	1944	1950	1969	1993	2016	2029	2042	2063	2068	2070	2090	2157
v	1482	1496	1511	1535	1540	1550	1551	1559	1576	1601	1604	1610
$k_{fractal}$	2166	2184	2208	2244	2250	2262	2268	2280	2304	2340	2346	2352
k_{old}	2166	2180	2195	2219	2224	2234	2235	2243	2260	2309	2312	2328
v	1624	1636	1640	1662	1673	1697	1706	1713	1721	1742	1762	1770
$k_{fractal}$	2376	2394	2400	2430	2448	2484	2496	2508	2520	2550	2574	2592
k_{old}	2352	2391	2400	2422	2433	2457	2466	2473	2481	2502	2524	2532

Table A.14: Further Improvements for Strength 9, Six Rows, Part 2

v	1781	1790	1802	1806	1811	1836	1843	1867	1880	1883	1891	1901
$k_{fractal}$	2604	2622	2640	2646	2652	2688	2700	2736	2754	2760	2772	2784
k_{old}	2558	2567	2637	2646	2651	2676	2683	2707	2720	2723	2731	2741
v	1906	1916	1944	1949	1964	1976	1980	1989	2018	2021	2031	2045
$k_{fractal}$	2790	2808	2850	2856	2880	2898	2904	2916	2958	2964	2976	3000
k_{old}	2746	2756	2784	2789	2804	2895	2904	2913	2942	2945	2955	2969
v	2061	2069	2087	2096	2098	2126	2135	2146	2158	2162	2175	2207
$k_{fractal}$	3024	3036	3060	3072	3078	3120	3132	3150	3168	3174	3192	3240
k_{old}	2985	2993	3027	3036	3048	3086	3095	3106	3165	3174	3187	3219
v	2231	2247	2252	2255	2272	2281	2288	2294	2316	2329	2336	2348
$k_{fractal}$	3276	3300	3306	3312	3330	3348	3360	3366	3402	3420	3432	3450
k_{old}	3243	3259	3264	3267	3284	3293	3300	3306	3328	3341	3348	3447
v	2352	2369	2401	2406	2425	2441	2449	2483	2486	2500		
$k_{fractal}$	3456	3480	3528	3534	3564	3588	3600	3648	3654	3672		
k_{old}	3456	3473	3505	3510	3529	3545	3553	3587	3590	3604		

Table A.15: Further Improvements for Strength 9, Seven Rows

v	479	494	503	515	527	539	557	566	575	593	599	614
$k_{fractal}$	1120	1155	1176	1204	1232	1260	1302	1323	1344	1386	1400	1435
k_{old}	711	742	760	779	810	833	857	866	875	893	899	923
v	623	638	650	662	683	686	701	710	731	740	758	767
$k_{fractal}$	1456	1491	1519	1547	1596	1603	1638	1659	1708	1729	1771	1792
k_{old}	935	950	962	974	1004	1014	1060	1074	1095	1104	1131	1149
v	773	797	863	899	971	1049	1064	1124	1142	1154	1172	1184
$k_{fractal}$	1806	1862	2016	2100	2268	2450	2485	2625	2667	2695	2737	2765
k_{old}	1157	1181	1299	1379	1511	1641	1656	1724	1742	1754	1781	1796
v	1214	1244	1259	1274	1289	1295	1307	1319	1322	1337	1343	1349
$k_{fractal}$	2835	2905	2940	2975	3010	3024	3052	3080	3087	3122	3136	3150
k_{old}	1838	1868	1883	1922	1967	1985	2063	2075	2078	2093	2099	2105
v	1352	1364	1385	1397	1415	1442	1457	1469	1499	1574	1649	1679
$k_{fractal}$	3157	3185	3234	3262	3304	3367	3402	3430	3500	3675	3850	3920
k_{old}	2114	2132	2153	2165	2183	2311	2326	2351	2381	2462	2549	2579
v	1754	1763	1835	1847	1874	1889	1949	2024	2057	2078	2183	2186
$k_{fractal}$	4095	4116	4284	4312	4375	4410	4550	4725	4802	4851	5096	5103
k_{old}	2654	2726	2915	2927	2954	2969	3029	3104	3153	3228	3527	3530
v	2204	2294	2339	2351	2447	2483						
$k_{fractal}$	5145	5355	5460	5488	5712	5796						
k_{old}	3548	3638	3683	3695	3791	3827						

Table A.16: Further Improvements for Strength 10, Six Rows

v	73	92	125	130	151	168	177	187	191	203	224	244
$k_{fractal}$	90	114	156	162	189	210	222	234	240	255	282	306
k_{old}	89	113	155	161	188	209	220	233	239	254	281	305
v	255	257	271	281	290	307	311	318	323	333	351	356
$k_{fractal}$	321	324	342	354	366	387	393	402	408	420	444	450
k_{old}	318	321	340	353	365	387	392	402	407	417	441	447
v	365	389	391	412	419	422	431	455	459	471	473	488
$k_{fractal}$	462	492	495	522	531	534	546	576	582	597	600	618
k_{old}	459	491	494	521	531	534	543	573	579	594	597	617
v	506	511	527	534	551	553	581	591	595	600	631	635
$k_{fractal}$	642	648	669	678	699	702	738	750	756	762	801	807
k_{old}	641	647	668	678	695	697	734	747	753	759	801	806
v	647	656	663	671	689	694	717	731	741	743	751	778
$k_{fractal}$	822	834	843	852	876	882	912	930	942	945	954	990
k_{old}	822	833	843	851	870	877	908	926	940	942	953	989
v	788	797	799	806	835	839	851	867	881	900	905	935
$k_{fractal}$	1002	1014	1017	1026	1062	1068	1083	1104	1122	1146	1152	1191
k_{old}	1002	1014	1016	1026	1059	1063	1078	1099	1118	1143	1150	1190
v	956	959	961	963	976	1003	1013	1022	1031	1045	1067	1071
$k_{fractal}$	1218	1221	1224	1227	1242	1278	1290	1302	1314	1332	1359	1365
k_{old}	1217	1221	1224	1227	1240	1272	1285	1297	1309	1328	1357	1362
v	1083	1106	1127	1134	1139	1144	1181	1205	1207	1209	1223	1229
$k_{fractal}$	1380	1410	1437	1446	1452	1458	1506	1536	1539	1542	1560	1566
k_{old}	1378	1409	1436	1446	1451	1456	1499	1531	1534	1537	1555	1563
v	1256	1266	1275	1291	1312	1319	1327	1331	1343	1373	1388	1401
$k_{fractal}$	1602	1614	1626	1647	1674	1683	1692	1698	1713	1752	1770	1788
k_{old}	1599	1613	1625	1646	1673	1683	1691	1695	1707	1745	1765	1782
v	1406	1411	1415	1455	1479	1481	1490	1511	1518	1537	1547	1556
$k_{fractal}$	1794	1800	1806	1857	1887	1890	1902	1929	1938	1962	1974	1986
k_{old}	1789	1796	1801	1854	1886	1889	1901	1928	1938	1957	1967	1979
v	1579	1607	1619	1621	1631	1668	1683	1701	1703	1706	1724	1731
$k_{fractal}$	2016	2052	2067	2070	2082	2130	2148	2172	2175	2178	2202	2211
k_{old}	2009	2046	2062	2065	2078	2127	2147	2171	2174	2178	2201	2211
v	1757	1781	1783	1799	1827	1841	1846	1856	1865	1895	1930	1935
$k_{fractal}$	2244	2274	2277	2298	2334	2352	2358	2370	2382	2421	2466	2472
k_{old}	2237	2266	2269	2290	2328	2346	2353	2366	2378	2418	2465	2471
v	1947	1951	1958	1991	2006	2024	2029	2033	2061	2075	2087	2111
$k_{fractal}$	2487	2493	2502	2544	2562	2586	2592	2598	2634	2652	2667	2697
k_{old}	2487	2492	2502	2535	2554	2578	2585	2590	2627	2646	2662	2694

Table A.17: Further Improvements for Strength 10, Six Rows, Part 2

v	2113	2136	2171	2183	2192	2199	2202	2239	2279	2281	2291	2309
$k_{fractal}$	2700	2730	2775	2790	2802	2811	2814	2862	2913	2916	2928	2952
k_{old}	2697	2727	2774	2790	2801	2811	2814	2854	2905	2908	2921	2945
v	2323	2342	2357	2375	2380	2391	2426	2445	2447	2454	2469	2471
$k_{fractal}$	2970	2994	3012	3036	3042	3057	3102	3126	3129	3138	3156	3159
k_{old}	2964	2989	3009	3033	3040	3054	3101	3126	3128	3138	3153	3155

Table A.18: Further Improvements for Strength 10, Seven Rows

v	161	251	269	287	419	449	479	539	749	767	791	809
$k_{fractal}$	217	338	367	393	572	621	673	748	1051	1073	1100	1129
k_{old}	215	335	355	388	563	605	644	725	1049	1067	1091	1109
v	839	863	899	1457								
$k_{fractal}$	1178	1213	1276	2107								
k_{old}	1149	1173	1220	2057								

Table A.19: Further Improvements for Strength 11, Seven Rows

v	23	35	47	53	59	71	83	89	95	107	119	125
$k_{fractal}$	29	44	59	67	74	90	104	113	121	136	152	159
k_{old}	27	41	56	63	71	87	101	110	117	132	148	156
v	149	161	167	197	215	233	251	263	269	287	293	299
$k_{fractal}$	191	205	214	251	277	297	324	338	347	371	379	386
k_{old}	187	202	210	247	273	293	314	329	341	366	375	383
v	311	323	329	335	359	377	383	389	395	407	419	431
$k_{fractal}$	400	418	425	434	465	489	497	503	512	524	544	560
k_{old}	395	407	413	422	459	484	493	501	507	519	532	549
v	449	461	467	479	485	503	527	539	545	569	575	587
$k_{fractal}$	581	599	606	623	631	654	686	702	709	737	749	764
k_{old}	569	593	601	618	627	647	671	683	693	731	743	760
v	593	599	611	623	629	647	659	671	683	701	713	719
$k_{fractal}$	773	781	794	812	819	844	860	875	888	915	929	939
k_{old}	768	777	791	803	809	827	839	851	873	909	926	934
v	725	755	767	779	791	809	815	827	839	857	863	881
$k_{fractal}$	947	986	1001	1018	1034	1057	1064	1076	1097	1121	1129	1149
k_{old}	943	975	989	1003	1015	1033	1043	1067	1091	1116	1125	1145
v	899	911	917	923	935	959	971	989	1007	1013	1019	1025
$k_{fractal}$	1176	1190	1199	1208	1224	1255	1270	1295	1319	1327	1334	1341
k_{old}	1163	1175	1181	1187	1203	1237	1257	1289	1314	1323	1331	1337
v	1049	1055	1079	1091	1103	1121	1133	1139	1151	1169	1175	1187
$k_{fractal}$	1369	1382	1414	1430	1442	1469	1483	1492	1509	1533	1541	1556
k_{old}	1361	1367	1391	1403	1415	1443	1467	1479	1503	1528	1537	1551

Table A.20: Further Improvements for Strength 11, Seven Rows, Part 2

v	1199	1223	1241	1247	1253	1259	1295	1319	1325	1343	1349	1367
$k_{fractal}$	1571	1604	1625	1636	1643	1652	1699	1730	1739	1763	1771	1794
k_{old}	1563	1587	1605	1611	1617	1623	1673	1721	1733	1758	1767	1787
v	1379	1385	1403	1427	1439	1451	1481	1499	1511	1517	1529	1535
$k_{fractal}$	1808	1817	1842	1874	1890	1904	1945	1966	1985	1991	2009	2017
k_{old}	1799	1805	1823	1855	1872	1889	1921	1955	1979	1987	2004	2013
v	1559	1583	1595	1619	1631	1637	1649	1655	1679	1709	1715	1727
$k_{fractal}$	2048	2079	2096	2128	2144	2151	2165	2174	2207	2247	2254	2271
k_{old}	2039	2063	2075	2099	2111	2117	2129	2135	2183	2241	2249	2266
v	1733	1763	1781	1793	1799	1823	1835	1847	1871	1889	1913	1919
$k_{fractal}$	2279	2318	2339	2357	2366	2398	2414	2429	2460	2485	2513	2525
k_{old}	2275	2307	2325	2337	2343	2367	2379	2391	2423	2459	2507	2519
v	1931	1937	1943	1949	1979	2015	2027	2039	2051	2069	2087	2099
$k_{fractal}$	2540	2549	2557	2563	2604	2652	2666	2684	2700	2723	2744	2762
k_{old}	2536	2544	2553	2561	2591	2632	2649	2663	2675	2693	2711	2729
v	2105	2111	2141	2159	2165	2183	2207	2231	2243	2249	2261	2267
$k_{fractal}$	2769	2779	2819	2843	2851	2873	2906	2934	2954	2961	2975	2986
k_{old}	2741	2753	2813	2838	2847	2867	2891	2915	2927	2933	2945	2951
v	2279	2303	2339	2345	2351	2375	2393	2399	2417	2429	2435	2447
$k_{fractal}$	3002	3033	3080	3089	3095	3129	3153	3161	3181	3199	3206	3224
k_{old}	2963	3007	3057	3066	3077	3123	3148	3157	3177	3189	3195	3207
v	2483	2495										
$k_{fractal}$	3272	3287										
k_{old}	3243	3255										

Table A.21: Further Improvements for Strength 11, Eight Rows

v	26	41	958	989	1006	1030	1054	1078	1116	1133	1150	1188
$k_{fractal}$	34	52	1600	1651	1680	1720	1760	1800	1862	1891	1920	1982
k_{old}	31	50	1314	1353	1370	1394	1469	1498	1548	1565	1582	1728
v	1198	1229	1246	1277	1301	1325	1370	1373	1404	1421	1466	1483
$k_{fractal}$	2000	2051	2080	2131	2171	2211	2284	2291	2342	2371	2444	2473
k_{old}	1738	1769	1786	1817	1841	1865	1942	1952	2004	2021	2066	2083
v	1521	1534	1552	1604	1726	1803	1942	2437				
$k_{fractal}$	2535	2561	2586	2670	2881	3006	3241	4060				
k_{old}	2121	2134	2152	2204	2486	2563	2824	3517				

Table A.22: Further Improvements for Strength 11, Nine Rows

v	29
$k_{fractal}$	39
k_{old}	36

Table A.23: Further Improvements for Strength 11, Ten Rows

v	32
$k_{fractal}$	44
k_{old}	41