8-2011

On the Complexity of Grid Coloring

Daniel Christopher Apon
University of Arkansas, Fayetteville

On the Complexity of Grid Coloring

On the Complexity of Grid Coloring

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science

By

Daniel Christopher Apon
University of Arkansas
Bachelor of Science in Business Administration, 2008

August 2011
University of Arkansas

Abstract

This thesis studies problems at the intersection of Ramsey-theoretic mathematics, computational complexity, and communication complexity. The prototypical example of such a problem is *Monochromatic-Rectangle-Free Grid Coloring*. In an instance of Monochromatic-Rectangle-Free Grid Coloring, we are given a chessboard-like grid graph of dimensions n and m, where the vertices of the graph correspond to squares in the chessboard, and a number of allowed colors, c. The goal is to assign one of the allowed colors to each vertex of the grid graph so that no four vertices arranged in an axis-parallel rectangle are colored monochromatically. Our results include:

1. A conditional, graph-theoretic proof that deciding Monochromatic-Rectangle-Free Grid Coloring requires time superpolynomial in the input size.

2. A natural interpretation of Monochromatic-Rectangle-Free Grid Coloring as a lower bound on the communication complexity of a cluster of related predicates.

3. Original, best-yet, *monochromatic-square-free* grid colorings: a 2-coloring of the 13×13 grid, and a 3-coloring of the 39×39 grid.

4. An empirically-validated computational plan to decide a particular instance of Monochromatic-Rectangle-Free Grid Coloring that has been heavily studied by the broader theory community, but remains unsolved: whether the 17×17 grid can be 4-colored without monochromatic rectangles. Our plan is based in high-performance computing and is expected to take one year to complete.

This thesis is approved for recommendation
to the Graduate Council.

Thesis Director:

Dr. Wing-Ning Li

Thesis Committee:

Dr. Gordon Beavers

Dr. Russell Deaton

Thesis Duplication Release

I hereby authorize the University of Arkansas Libraries to duplicate this thesis when needed for research and/or scholarship.

Agreed

Daniel Christopher Apon

Refused

Daniel Christopher Apon

Acknowledgements

I am profoundly indebted to all of those in my life, whose constant support and encouragement have made this thesis possible.

To Dr. Li, who carved out time for an individual independent study course to enable me to explore all kinds of wild topics in theory, who has always been available to meet with me, whose teaching and emphasis on precision of thought has made me into the computer scientist I am today–

To the outstanding computer science department at the University of Arkansas that has supported me these past years with a Distinguished Doctoral Fellowship–

To my father, my brother, my friends, and God, who anchor my life–

And to my mother, for the countless hours spent listening to me ramble aimlessly over the phone about math, for the lunches you brought me when I was stuck in the office working, and for the example you set for me academically–

Thank you.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Shape-Free Grid Coloring

This thesis studies the computational complexity of a family of problems, *Shape-Free Grid Coloring*, and applied algorithmic approaches to solving specific instances of problems in the family where overwhelming evidence suggests they are computationally intractable. Shape-Free Grid Coloring is a family of related problems arising in Ramsey-theoretic mathematics that are distinguished from one another based on a given geometric pattern. The prototypical example of such a problem is *Monochromatic-Rectangle-Free Grid Coloring*. In an instance of Monochromatic-Rectangle-Free Grid Coloring, we are given a chessboard-like grid graph of dimensions n and m, where the vertices of the graph correspond to squares in the chessboard, and a number of allowed colors, c. The goal is to assign a color in the set $\{0, 1, ..., c-1\}$ to each vertex (i, j) for $0 \leq i < n$ and $0 \leq j < m$ such that there exist no *monochromatic rectangles*. A monochromatic rectangle is a set of four vertices arranged in a rectangular shape that are assigned the same color, i.e. $\{(i, j), (i+a, j), (i, j+b), (i+a, j+b)\}$ for integers a, b where

- $-i \leq a < n - i$,
- $-j \leq b < m - j$,
- $a \neq 0$, and
- $b \neq 0$.

We begin with a simple, concrete example of an instance of Monochromatic-Rectangle-Free Grid 2-Coloring. As shown below (Fig. 1.1), we have two 4x4 grid

graphs that are assigned colors out of $\{0, 1\}$, where 0 is red and 1 is blue. The left grid is *legally* colored, since it contains no monochromatic rectangles. The right grid is *illegally* colored; the monochromatic rectangle induced by the color assignment is marked with $\times$'s.

As discussed above, an instance of Monochromatic-Rectangle-Free Grid Coloring is specified by the input parameters n, m, and c. The output of an algorithm that solves Monochromatic-Rectangle-Free Grid Coloring is a `YES` or `NO` answer to whether it is possible to color such a grid without forming a monochromatic rectangle, and the witness to the output is the coloring itself. In general however, Shape-Free Grid Coloring is a family of closely related computational decision problems, where in each problem description we specify the geometric structure that we disallow (e.g., a monochromatic rectangle) and in the problem input we specify the colorable space (e.g., a grid's dimensions) and the number of allowed colors (i.e., the parameter c).

In the following three sections, we give a brief background on Ramsey Theory, discuss the motivations for studying Shape-Free Grid Coloring, and survey our results.

1.2 Ramsey Theory

An essential property of Shape-Free Grid Coloring that sets it apart from typical graph coloring problems in computational complexity is that Ramsey's theorem and

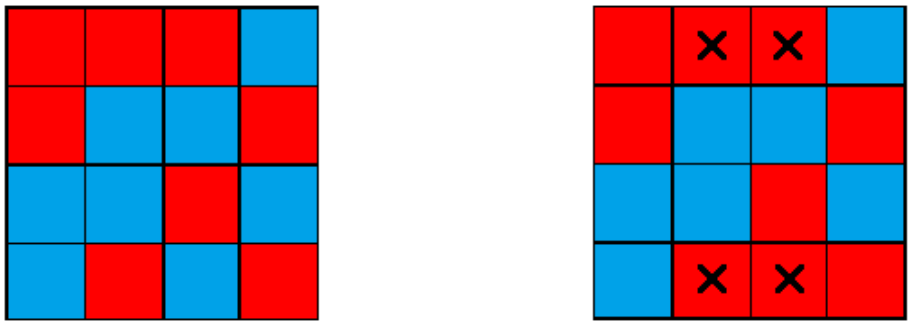

Figure 1.1: A legally 2-colored grid, and an illegally 2-colored grid

generalizations thereof apply to the family of problems. Ramsey's theorem states:

Ramsey's Theorem (restated)([36]). *For any given integer c, and any given integers $n_1, ..., n_c$, there exists some number, $R(n_1, ..., n_c)$, such that if the edges of a complete graph of order $R(n_1, ..., n_c)$ are colored with c different colors, then for some i between 1 and c, there must exist a complete subgraph of order n_i whose edges are all color i.*

To gain some intuition about how Ramsey's theorem operates, consider the *pigeonhole principle.* That is, if n pigeons are placed inside m pigeonholes, how big must n be before we are guaranteed that at least one pigeonhole holds at least two pigeons? The pigeonhole principle says if $n > m$, this property holds by a simple counting argument.

Ramsey's theorem, then, is a classic generalization of the pigeonhole principle for graph and number theory. In other words, if we have a large enough complete graph colored with some constant number of colors, we are guaranteed that there exists a complete monochromatic subgraph. In fact, the underlying theme of Ramsey-theoretic mathematics asks questions of the form: "Given some mathematical structure, how large must it be to guarantee that a particular property will hold?"

In particular, the generalization of Ramsey's theorem that directly applies to Shape-Free Grid Coloring is Van der Waerden's theorem:

Van der Waerden's Theorem (restated)([40]). *For any given positive integers r and k, there is some number N such that if the integers $\{1, 2, \ldots, N\}$ are colored, each with one of r different colors, then there are at least k integers in arithmetic progression all of the same color.*

Further, an *arithmetic progression* is a sequence of numbers such that the difference of any two successive members of the sequence is a constant. Implicit throughout this thesis are projections from a sufficiently large, contiguous subset of the natural numbers to finite, discrete geometric regions, such as grids, so that Van der Waerden's Theorem applies.

Thus, in the case of Shape-Free Grid Coloring, in constrast to traditional graph coloring problems, the underlying question is often *not* "Can this graph be legally colored?" but rather "How big can this graph be and still be legally colored?" In fact, a simple observation along these lines is that any n-by-m grid graph can always be colored without monochromatic rectangles if we are allowed at least $\min(n, m)$ colors, by simply assigning a different color to each row (resp. column). If, however, we fix some constant number of colors c and allow n and m to grow together linearly (or for large enough n, we allow m to grow, and vice versa), we are guaranteed that at some point the graph will no longer be legally colorable, for any reasonable and fixed geometric shape. Thus, for any fixed number of colors c and beyond some threshold of n and m, there is an infinite range of `NO` instances and a finite number of `YES` instances. Many of these `YES` instances are trivially solvable, while others – right at the boundary between `YES`'s and `NO`'s – appear to be quite difficult to resolve.

As an aside for later, the existence of a *finite* number of `YES` instances for fixed c and sufficiently large n and m, as n and m grow, plays a key role in the difficulties associated with classifying the complexity of Shape-Free Grid Coloring.

1.3 Motivations

In the following section, we motivate the two main chapters of this thesis. The first part explores a protocol problem in communication complexity from where the question of the complexity of problems in Shape-Free Grid Coloring arises. The second part surveys previous, related work and introduces the 17x17 4-Color Problem.

1.3.1 Communication Complexity Foundations

In this part, we introduce the communication complexity model of [29] and reproduce an abbreviated communication protocol lower bound proof and discussion from [11], due to its high degree of relevance and intimacy with the Shape-Free Grid Coloring problem.

In the typical *two-party* communication complexity model, we are given arbitrary finite sets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ and let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be an arbitrary function. There are two players, Alice and Bob, who wish to evaluate $f(x, y)$, for some inputs $x \in X$ and $y \in Y$. However, Alice only knows x and Bob only knows y. Thus, to evaluate the function, Alice and Bob will communicate with one another by sending a single bit at a time. Importantly, the communication is carried out according to some protocol $\mathcal{P}$ that is fixed ahead of time and which depends only on f. At each stage of the protocol $\mathcal{P}$, the players (abiding by the protocol) determine whether the run terminates by specifying the output $f(x, y)$.

In particular, we are only interested in the *amount* of bits communicated, regardless of the internal computations required of Alice and Bob. Therefore, we treat Alice and Bob as if they have *unlimited computational power*. The *cost of a protocol $\mathcal{P}$ on input* (x, y) is the number of bits communicated by $\mathcal{P}$ on input (x, y). The *cost (or complexity) of a protocol* $\mathcal{P}$ is the worst case, or maximal, cost of $\mathcal{P}$ over all inputs (x, y). The *communication complexity of* f is the minimum cost over all protocols

that compute f.

Now, we generalize this model. Assume there are k processes (or players) $P_0, ..., P_{k-1}$ and k integers $a_0, ..., a_{k-1} \in \{0, ..., n\}$ for some positive integer n. Each process has access to all the a_j, except that P_i is denied access to a_i. Finally, all bits transmitted from one process to another are sent through a broadcast channel that all processes can freely monitor. This model is commonly referred to as the *Number-On-Forehead (NOF) Multi-Party Communication Complexity Model*.

We define a *broadcast history* $b \in (0+1)^*$ to be a record of all bits transmitted by all processes up to a certain point in time. A *k-party protocol* is a deterministic algorithm running on each process P_i that determines, from the numbers P_i knows and the broadcast history, what bit P_i should transmit at times $i, i+k, ...$ That is, communication occurs in a purely cyclic fashion: at time $t = 0$, P_0 examines the information it has access to and broadcasts one bit; at time $t = 1$, P_1 does the same, and so on.

Let $H_n = \{0, ..., n\}^k$ be a k-dimensional hypercube, where each vector $\langle v_0, v_1, ..., v_{k-1} \rangle \in H_n$ describes a *situation*, or an assignment to the variables $a_0, ..., a_{k-1}$. Observe that the (discrete, finite) vector space in H_n describes all possible inputs to an instance of the NOF model. Then, let $\mathcal{P}$ denote a k-party protocol where each party P_i is given inputs $a_j \in \{0, ..., n\}$. Each of the concrete combinations of possible a_i specify some situation $\bar{v} \in H_n$, and for each $\bar{v} \in H_n$, $\mathcal{P}$ uniquely determines a string $\mathcal{P}(\bar{v}) \in (0+1)^*$ that is the complete broadcast history for the k processes in situation $\bar{v}$. In general, we will abuse terminology and use the $\mathcal{P}(\cdot)$ notation interchangably to refer either to a protocol by which the parties compute a predicate or a function that assigns broadcast histories to H_n.

Finally, we provide a definition of protocol validity. For simplicity, we will say a k-party protocol $\mathcal{P}$ is *valid* for a predicate Q if and only if it can be used by processes $P_0, ..., P_{k-1}$ to decide Q. By *decide*, we mean that the parties will communicate

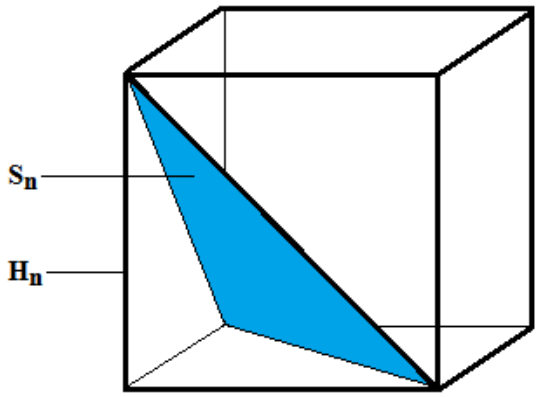

Figure 1.2: The plane of solutions, S_n, circumscibed by the situation space, H_n

according to the protocol $\mathcal{P}$, and at some prearranged time that depends on $\mathcal{P}$, the parties must all halt with either all *correctly accepting* or all *correctly rejecting* (that is, either all output `YES` or all output `NO`, and the joint output must match the definition of the predicate, given the specific instance of the parties' inputs). Note that a sufficient and necessary condition for a protocol $\mathcal{P}$ to be *invalid* is for any single process P_i to incorrectly accept *or* reject on *any* point $\bar{v} \in H_n$.

Now let the predicate *Exactly-n* be true for k integers $a_0, ..., a_{k-1}$ if and only if $a_0 + \cdots + a_{k-1} = n$. Now we prove the following lower bound for the number of bits that must be communicated for k parties to compute *Exactly-n*:

Theorem 1. *Let the solution "plane" of Exactly-n be* $S_n = \{\bar{v} \in H_n : \sum_i v_i = n\}$. *The complexity of any k-party protocol for Exactly-n is bounded below by the logarithm of the minimum number of colors required to color the points of* S_n *so that no "forbidden k-patterns" are colored monochromatically.*

The proof will proceed as follows. Let H_n be the set of situations. Let S_n be the hyperplane on which the processes must "accept" the input. A three-dimensional geometric interpretation of this set-up is shown in Fig. 1.2 above. In general, we

say that the number of bits transmitted by a protocol determines *how large* a set of potential, distinct broadcast histories may exist. We will show that if a protocol determines too few distinct broadcast histories on the plane S_n, then it is not valid for *Exactly-n*.

We define a *forbidden k-pattern* as a set of k distinct points $\bar{v}_0, ..., \bar{v}_{k-1} \in H_n$ if there is a point $\bar{w} \in H_n$ such that, for each i, $\bar{w}$ differs from $\bar{v}_i$ only in coordinate i. Note that for the case $k = 3$ for *Exactly-n*, forbidden k-patterns on the plane S_n are equilateral triangles.

Here is an explicit example for the case $k = 3$, $n = 2$. First, we enumerate all of the triples of integers $\langle v_0, v_1, v_2 \rangle$ for $v_i \in [0, 2]$ that sum precisely to 2:

$$S_3 = \{\langle 2,0,0\rangle, \langle 0,2,0\rangle, \langle 0,0,2\rangle, \langle 1,1,0\rangle, \langle 1,0,1\rangle, \langle 0,1,1\rangle\}$$

Then we group them[1] into every possible forbidden 3-pattern, generate the resulting $\bar{w}$ by the definition of a forbidden k-pattern, and observe that $\sum_K v_{i,j,K} = 2 = n$ and $\sum_j w_{i,j} \neq 2 = n$.

$$(\{\bar{v}_{0,0} = \langle 2,0,0\rangle, \bar{v}_{0,1} = \langle 0,2,0\rangle, \bar{v}_{0,2} = \langle 0,0,2\rangle\}, \bar{w}_0 = \langle 0,0,0\rangle)$$

$$(\{\bar{v}_{1,0} = \langle 2,0,0\rangle, \bar{v}_{1,1} = \langle 1,1,0\rangle, \bar{v}_{1,2} = \langle 1,0,1\rangle\}, \bar{w}_1 = \langle 1,0,0\rangle)$$

$$(\{\bar{v}_{2,0} = \langle 1,1,0\rangle, \bar{v}_{2,1} = \langle 0,2,0\rangle, \bar{v}_{2,2} = \langle 0,1,1\rangle\}, \bar{w}_2 = \langle 0,1,0\rangle)$$

$$(\{\bar{v}_{3,0} = \langle 1,0,1\rangle, \bar{v}_{3,1} = \langle 0,1,1\rangle, \bar{v}_{3,2} = \langle 0,0,2\rangle\}, \bar{w}_3 = \langle 0,0,1\rangle)$$

$$(\{\bar{v}_{4,0} = \langle 0,1,1\rangle, \bar{v}_{4,1} = \langle 1,0,1\rangle, \bar{v}_{4,2} = \langle 1,1,0\rangle\}, \bar{w}_4 = \langle 1,1,1\rangle)$$

[1]Regarding notation throughout this thesis, when specifically referring to groups of forbidden k-patterns, we will generally use the integer i to refer to the index of a (*forbidden k-pattern*, $\bar{w}$) pair in the group, the integer j to refer to the index of the vectors $\bar{v}_i$ within a forbidden k-pattern and the indices of the coordinates of the $\bar{w}_i$, and the integer K to refer to the indices of the coordinates of the $\bar{v}_{i,j}$. Take care to avoid confusing the number of parties k and the indices K of individual coordinates $v_{i,j,K}$.

Lemma 1.1. *Let $\mathcal{P}$ be a protocol for processes $P_0, ..., P_{k-1}$. Let $H_n = \{0, ..., n\}^k$ be the hypercube of situations, and let $\bar{w} = \langle w_0, ..., w_{k-1} \rangle$ be any point in H_n. If $\bar{v}_0, ..., \bar{v}_{k-1}$ is a forbidden k-pattern for $\bar{w}$ and the complete broadcast histories at the $\bar{v}_j$ are identical, that is*

$$\alpha = \mathcal{P}(\bar{v}_0) = \mathcal{P}(\bar{v}_1) = \cdots = \mathcal{P}(\bar{v}_{k-1}), \tag{1.1}$$

then $\mathcal{P}(\bar{w}) = \alpha$.

Proof. We prove the lemma by induction on the length of the broadcast history, that for all i, the i^{th} bit broadcast at $\bar{w}$ is the same as the i^{th} bit broadcast at $\bar{v}_{i(\text{mod } k)}$.

Base step: If $|\alpha| = 0$, then every process immediately halts on $\bar{w}$ since it immediately halts on $\bar{v}_i$.

Inductive step: Assume that for all histories of length $< t$, the inductive hypothesis is true. Consider the t^{th} bit to be broadcast. By induction, $P_{t(\text{mod } k)}$ sees the same broadcast history at $\bar{w}$ and at $\bar{v}_{t(\text{mod } k)}$. Therefore, $P_{t(\text{mod } k)}$ broadcasts the same bit at time t at $\bar{w}$ as it would at time t at $\bar{v}_{t(\text{mod } k)}$. ■

Further note that Lemma 1.1 holds in the case of invalid protocols as well as valid ones.

Lemma 1.2. *A k-party protocol $\mathcal{P}$ is not valid for the predicate Exactly-n if it assigns the same broadcast history to all points of a forbidden k-pattern of S_n.*

Proof. Suppose $\bar{v}_0, ..., \bar{v}_{k-1}$ is a forbidden k-pattern on S_n for the point $\bar{w}$, such that the equality relationship in Equation (1.1) holds. By geometry, $\bar{w}$ is not on S_n. By Lemma 1.1, $\mathcal{P}(\bar{w}) = \alpha$. Therefore, $\mathcal{P}$ is not valid for *Exactly-n*. ■

Define the integer $\chi_k(n)$ to be the smallest number of colors required to color the points of S_n so that no forbidden k-pattern on S_n is colored monochromatically. Now we can prove Theorem 1.

Proof of Theorem 1. Consider a protocol $\mathcal{P}$ that computes *Exactly-n*. Let each distinct broadcast history $\mathcal{P}(\bar{v})$, for $\bar{v} \in H_n$, define a color. (Note that this colors all the points of S_n.) If some forbidden k-pattern on S_n is colored monochromatically, then by Lemma 1.2, $\mathcal{P}$ is not valid for *Exactly-n*, which is a contradiction. Therefore, there must be more than $\chi_k(n)$ distinct broadcast histories, and hence in some situation, $\Omega(\log(\chi_k(n)))$ bits must be communicated. ■

We point out that the choice of predicate in Theorem 1 (i.e. *Exactly-n*) is somewhat arbitrary. In fact, for *any* choice of predicate in the family *Exactly-m* for positive integers $m \in [0, kn]$, the subsequent proof will proceed exactly as above with minimal modification. Further, the authors continue on to show that the lower bound of Theorem 1 in optimal, up to an additive constant, by providing a matching upper bound by developing a k-party protocol that is valid for *Exactly-n* that costs $\log(\chi_k(n)) + k$ bits and that crucially uses the value of $\chi_k(n)$ for the protocol (in fact, the parties exchange information about their evaluation of $\chi_k(n)$).

Finally, in [11], the exact value of $\chi_k(n)$ is left unknown, relying on the parties' unbounded computational power to determine it. Indeed, the authors simply state, "the optimal protocol can be implemented after an exhaustive search." This begs the question of whether an exhaustive search is computationally optimal. Put more formally, one might ask, "What is the (computational) complexity of computing $\chi_k(n)$?"

1.3.2 Previous work

The applied segment of our work primarily picks up from the paper "Rectangle Free Coloring of Grids" by Fenner, Gasarch, Glover, and Purewal[13]. In this paper, the question asked is "what are the exact values of m and n for which $G_{n,m}$ is c-colorable?" $G_{n,m}$ denotes the n-by-m grid graph, and c-colorable refers to the existence of a monochromatic-rectangle-free c-coloring of $G_{n,m}$. In [13], an *obstruction set* is defined as the set of grids $G_{n,m}$ such that, for a fixed c, the grids are not c-colorable but all grids properly contained within this set of grids are c-colorable. In essence, obstruction sets refer to the aggregate threshold of grid sizes at which the grids cease being colorable.

Following their notation, we will formally define OBS_c as the following:

Fix c. Then OBS_c is the set of all grids $G_{n,m}$ such that $G_{n,m}$ is not c-colorable but all grids properly contained in $G_{n,m}$ are c-colorable. Such grids are also called *c-minimal*.

To briefly summarize a major part of their results, they prove the following:

- $\text{OBS}_2 = \{3\text{x}7, 5\text{x}5, 7\text{x}3\}$,
- $\text{OBS}_3 = \{19\text{x}4, 16\text{x}5, 13\text{x}7, 11\text{x}10, 10\text{x}11, 7\text{x}13, 5\text{x}16, 4\text{x}19\}$,
- $\text{OBS}_4 \supset \{41\text{x}5, 31\text{x}6, 29\text{x}7, 25\text{x}9, 23\text{x}10, 10\text{x}23, 9\text{x}25, 7\text{x}29, 6\text{x}31, 5\text{x}41\}$,
- In addition, OBS_4 contains exactly one of: 21x13, 21x12, and
- OBS_4 contains exactly one of: 19x17, 18x17, 17x17.
- Finally, if $19\text{x}17 \subset \text{OBS}_4$, then 18x18 could be in OBS_4 as well.

A particular instance of the remaining, unknown sets in OBS_4 – the 17x17 – has received a large amount of public attention in the TCS and puzzle-solving communities. (See [15][16][26][9][23][24].) I have posed the question on the TCS StackExchange Q&A site[6]. In fact, there has even been a web-based, HTML5 game modeled after the problem[37].

We formally define the problem as:

The 17x17 4-Color Problem. A *rectangle of* $G_{17,17}$ is a set of the form $\{(i,j),(i+a,j),(i,j+b),(i+a,j+b)\}$ for constants i, j, a, b such that the rectangle is contained in $G_{17,17}$. Given four colors, is there a way to color every element of $G_{17,17}$ such that $G_{17,17}$ does not contain a rectangle of all the same color?

Previous progress on the 17x17 4-Color Problem falls into two categories. The first comes from an unpublished manuscript by Elizabeth Kupin from Rutgers[28]. In the manuscript, Kupin identifies the existence of two, unique, monochromatic-rectangle-free, partial, 1-color subsets of $G_{17,17}$ by an exhaustive counting argument (that is, a full assignment of one color, with the remaining cells left uncolored). In other words, if $G_{17,17}$ is in fact legally 4-colorable, the final coloring must contain one of Kupin's partial colorings (up to the permutation of rows and columns of the grid).

The second main area of progress is in *approximate* colorings of $G_{17,17}$. Approximate colorings are defined as illegal colorings of $G_{17,17}$ (i.e. those that contain at least one monochromatic rectangle) and are ranked on how many monochromatic rectangle constraints are violated. A leaderboard of best approximations to a solution (supposing one exists) for the 17x17 4-Color Problem is maintained online[39].

1.4 Our results

1.4.1 Theoretical results

Our first, highlighted, theoretical result is a conditional complexity classification of Monochromatic-Rectangle-Free Grid Coloring.

To set up the statement of the theorem, we begin with an informal version of the Exponential Time Hypothesis (ETH)[25]:

Exponential Time Hypothesis. 3-SAT *cannot be solved in subexponential time in the worst case.*

We further remark that the ETH is widely believe to hold and is equivalent to the conjecture, FPT $\neq$ W[1], a sister to P $\neq$ NP in the world of parameterized complexity theory. That is, due to [2], it is known that FPT = W[1] implies that SAT is in **DTIME**$(2^{o(n)})$.

Informally, a *constraint satisfaction problem* (CSP) is a generalization of SAT allowing more than two, discrete truth values per variable; a formal definition follows in Chapter 2. In order to defer a lengthy technical discussion to Chapter 2, we state an informal version of the assumption we require:

The NAE is Robust for Grids Assumption. *Not-All-Equal relations can encode any* CSP *relation over grid-shaped hypergraphs.*

The intuition for this assumption fundamentally comes from the manner in which the NAE-SAT problem structure can be used to properly encode SAT instances. In essence, we draw on the intuition of the SAT $\leq_P$ NAE-SAT NP-completeness reduction.

Then in the first half of Chapter 2, we provide evidence that Monochromatic-Rectangle-Free Grid Coloring is not tractable:

Theorem 12. (Informal) *Assume the Exponential Time Hypothesis and the NAE is Robust for Grids Assumption. Then, given a partially complete coloring of $G_{n,m}$, it requires superpolynomial time in the worst case to decide whether that coloring can be extended to a complete, valid monochromatic-rectangle-free coloring of $G_{n,m}$.*

Another interpretation of this theorem is that it highlights another candidate-route for classifying the complexity of Monochromatic-Rectangle-Free Grid Coloring. That is, since efforts from across the broader theory community have been unable to show that the problem is NP-complete (and yet it seems unlikely that the problem is in P), it may be fruitful to look for other problems, similarly not known to be NP-complete, that share properties with Monochromatic-Rectangle-Free Grid Coloring.

We then report on the results of a search for applications of Monochromatic-Rectangle-Free Grid Coloring to other areas of theoretical computer science. In particular, we demonstrate that Monochromatic-Rectangle-Free Grid Coloring provides lower bounds in multiparty communication complexity:

Main Theorem. *Instances of Monochromatic-Rectangle-Free Grid Coloring that are c-minimal provide lower bounds for the multiparty communication complexity of the predicates Exactly-$\frac{k}{2}n$, PlanarExactly-n, and LinearExactly-n for the 4 party case.*

Finally, based on this discovery and similar evidence in the literature, cited in Chapter 1, we conjecture that this relationship extends further. In particular, we define *Shape-Free Space Coloring* as an analog of Shape-Free Grid Coloring for finite, discrete, d-dimensional spaces for $d \geq 3$ and conjecture the following:

The Ramsey-Communication Conjecture. *Fix any problem in Shape-Free Space Coloring. Then c-minimal instances of this problem provide lower bounds on the multiparty communication complexity of some predicate.*

1.4.2 Applied results

Despite our belief that Monochromatic-Rectangle-Free Grid Coloring is inherently intractable, in Chapter 3 we demonstrate some progress towards answering the 17x17 4-Color Problem. As a "warm up," we show a legal, *monochromatic-square-free* 2-coloring of $G_{13,13}$ as well as a legal, *monochromatic-square-free* 3-coloring of $G_{39,39}$. While neither result is known to be optimal for square-free colorings, both are surprising. In the square-free, 2-color case, it has been believed that $G_{10,10}$ was the threshold for monochromatic-square-free 2-colorable grids[41]. In the square-free, 3-color case, a simple calculation shows that the candidate solution space for monochromatic-square-free 3-colorings of $G_{39,39}$ massively dwarfs the candidate solution space for monochromatic-rectangle-free 4-colorings of $G_{17,17}$. In particular, $3^{39\times 39} = 3^{1521} > 10^{725} \gg 10^{174} > 4^{289} = 4^{17\times 17}$.

We then use the intuition gained from finding monochromatic-square-free colorings to design an algorithmic approach based in high-performance computing to answer the 17x17 4-Color Problem. In particular, we develop a plan based on empirical testing that is expected to solve the 17x17 4-Color Problem within *one year*. Key to our approach is our hypothesis that *no* coloring of $G_{17,17}$ results in a monochromatic-rectangle-free 4-coloring. That is, we believe it is crucial to focus on investigating approaches that allow the existence of a solution to the 17x17 4-Color Problem to be falsified. Taken together with the theoretical evidence of Chapter 2, this suggests the optimal approach will likely involve a brute-force search in some form. Indeed, our technique has, at its core, an exhaustive, backtracking-style algorithm.

Chapter 2

On the Complexity of Grid Coloring

2.1 Preliminaries

We assume basic familiarity with polynomial- and exponential-time algorithms, computational complexity theory (e.g. Big-Oh notation), graph theory (e.g. the concepts of vertices, edges, and graphs), and simple first-order logic (e.g. the definition of a relation). We also assume familiarity with basic complexity classes, e.g. P and NP. More specialized concepts used in this chapter from complexity theory (e.g. constraint satisfaction problems) and graph theory (e.g. hypergraphs and tree decompositions) are briefly described below. The classic computational complexity textbook of Papadimitriou[33] or the textbook of Arora and Barak[8] cover all of the requisite complexity concepts, with an especially excellent series of chapters devoted to first-order logic in the former. The graph theory textbook of Gross and Yellen[21] contains an excellent introduction to basic graph theory as well as a chapter discussing the foundations of Ramsey Theory.

2.1.1 Definitions

A *constraint satisfaction problem* (CSP) is a generalization of SAT, allowing more than two truth values. An instance of a CSP is specified by a triple $I = (V, C, D)$ where V is a set of *variables*, D is a set of values that the variables may take called the *domain*, and C is a set of *constraints* of the form $\langle (v_1, \ldots, v_k), R \rangle$, where $k \geq 1$ and R is a k-ary relation on D. A *solution* to some instance I is an assignment $\alpha : V \to D$ such that for all constraints $\langle (v_1, \ldots, v_k), R \rangle \in C$, we have $(\alpha(v_1), \ldots, \alpha(v_k)) \in R$.

Constraints may be specified in either of the following two manners: (1) by explicitly enumerating all of the possible combinations of values of variables that are

legal (or, illegal); that is, all of the tuples of R, (2) by implicitly representing the set of constraints by a set of mathematical or logical properties. Examples of the latter, which will be implicitly used throughout this thesis, include CNF formulas in SAT instances. As an aside, we remark that this can result in an exponentially more succinct representation of problem instances, and often, it is important to use such a representation to avoid misrepresenting the actual size of the input problem.

For example, consider the following (*very* simple) CSP instance of the first form. We let $V = \{x_1, x_2, x_3\}$, $D = \{0, 1, 2\}$, and $C = \{\langle (x_1, x_2, x_3), R\}$ where $R = \{(0,0,1), (0,0,2), (0,1,0), (0,1,1), (0,1,2), (0,2,0), (0,2,1), (0,2,2), (1,0,0), (1,0,1), (1,0,2), (1,1,0), (1,1,2), (1,2,0), (1,2,1), (1,2,2), (2,0,0), (2,0,1), (2,0,2), (2,1,0), (2,1,1), (2,1,2), (2,2,0), (2,2,1)\}$. Now, consider the situation of the same CSP in the second form. Let NAE_k be the k-ary relation such that the variables the relation is between are not all equal. Then we write the previous CSP instance as $V = \{x_1, x_2, x_3\}$, $D = \{0, 1, 2\}$, and $C = \{\langle (x_1, x_2, x_3), NAE_3\rangle\}$.

An (undirected) *graph*, as is standard, is a pair $G = (V, E)$, consisting of a set V of *vertices* and a set of *edges* E of unordered pairs of $v \in V$. A *hypergraph* is a pair $H = (V(H), E(H))$, consisting of a set $V(H)$ of vertices and a set $E(H)$ of subsets of $V(H)$, the *hyperedges* of H. We note that *by definition* all graphs are hypergraphs, though not all hypergraphs are graphs. Here is an example of a hypergraph on five vertices with three hyperedges: Let $H_0 = (V(H_0), E(H_0))$ and $V(H_0) = \{x_1, x_2, x_3, x_4, x_5\}$ and $E(H_0) = \{\{x_1, x_2, x_3\}, \{x_2, x_3, x_4\}, \{x_1, x_5\}\}$.

A *tree decomposition* of a hypergraph H is a tuple $(T, (B_t)_{t \in V(T)})$, where T is a tree and $(B_t)_{t \in V(T)}$ is a family of subsets of $V(H)$ such that for each $e \in E(H)$ there is a node $t \in V(T)$ such that $e \subseteq B_t$ and for each $v \in V(H)$ the set $\{t \in V(T) : v \in B_t\}$ is connected in T. The sets B_t are called the *bags* of the decomposition. Finally, the *width* of a tree decomposition $(T, (B_t)_{t \in V(T)})$ is $\max\{|B_t| : t \in V(t)\} - 1$. The *treewidth*, $\mathrm{tw}(H)$, of a hypergraph H is the minimum width over all tree decomposi-

tions of H.

Here are two (not necessarily optimal) examples of the construction of the tree and its bags for a tree decomposition of the previous hypergraph, H_0: We let the leaves of the tree T_{H_0} be labeled by the hyperedges of H_0. That is, there are three leaves labeled with $\{x_1, x_2, x_3\}$, $\{x_2, x_3, x_4\}$, and $\{x_1, x_5\}$ respectively. We let there be a parent bag of the bags labeled $\{x_1, x_2, x_3\}$ and $\{x_2, x_3, x_4\}$ that is labeled with $\{x_1, x_2, x_3, x_4\}$. Finally, there is a root bag, the parent of $\{x_1, x_2, x_3, x_4\}$ and $\{x_1, x_5\}$, that is labeled $\{x_1\}$. Observe that this tree decomposition has width 3. Further, there is a trivial, alternate tree decomposition of H_0, where the tree T'_{H_0} has a single bag labeled $\{x_1, x_2, x_3, x_4, x_5\}$. This tree decomposition has width 4.

We say that a class $\mathcal{H}$ of hypergraphs has *bounded treewidth* if there exists an integer k such that $\text{tw}(H) \leq k$ for all $H \in \mathcal{H}$; if no such integer exists, then we say $\mathcal{H}$ has *unbounded treewidth*. For a class $\mathcal{H}$ of hypergraphs, we say $\mathcal{H}$ has *bounded hyperedge size* if there exists an integer k such that for all hypergraphs $H = (V(H), E(H)) \in \mathcal{H}$ and for all $e \in E(H), |e| \leq k$.

For a CSP instance $I = (V, D, C)$ and a hypergraph H, if V is the vertex set of H and for every constraint in C, there is precisely one hyperedge in H that consists of all variables occuring in the constraint, we say H *is the hypergraph of the* CSP *instance* I. For a hypergraph H, we denote the set of all instances I where H is the hypergraph of I by $\mathsf{CSP}(H)$. For a class of hypergraphs $\mathcal{H}$, we denote by $\mathsf{CSP}(\mathcal{H})$ the union of all $\mathsf{CSP}(H)$ for hypergraphs H in $\mathcal{H}$. That is, we say $\mathsf{CSP}(\mathcal{H}) = \bigcup_{H \in \mathcal{H}} \mathsf{CSP}(H)$.

2.1.2 The relationship between CSPs and treewidth

The following series of theorems form the core basis of the main proof in the first half of this chapter. The first, due to [19] and [18], is restated here:

Theorem 2 (restated) ([19], [18]). *Assume the ETH. Let $\mathcal{C}$ be a family of graphs. Then, the class of instances $\mathsf{CSP}(\mathcal{C})$ is polynomial-time-solvable in the worst case* if and only if *$\mathcal{C}$ has bounded treewidth.*

We further remark that the Exponential Time Hypothesis (ETH)[25] is widely believe to hold and is equivalent to the conjecture, $\mathsf{FPT} \neq \mathsf{W}[1]$, a sister to $\mathsf{P} \neq \mathsf{NP}$ in the world of parameterized complexity theory. That is, due to [2], it is known that $\mathsf{FPT} = \mathsf{W}[1]$ implies that SAT is in deterministic time $2^{o(n)}$.

Due to [20], the result of Theorem 2 can be generalized to $\mathsf{CSP}(\mathcal{H})$ for classes $\mathcal{H}$ of hypergraphs of bounded hyperedge size. That is, we have the following powerful theorem:

Theorem 3 (restated) ([20]). *Assume the ETH. Let $\mathcal{H}$ be a family of hypergraphs of bounded hyperedge size. Then we have the following implications:*

$$\mathsf{CSP}(\mathcal{H}) \in \textbf{PTIME} \Longleftrightarrow \mathcal{H} \textit{ has bounded treewidth.}$$

2.1.3 The relationship between brambles and treewidth

We provide a few definitions required for the final graph-theoretic notion we require. Given a graph $G = (V, E)$, we say two subsets of V *touch* if (1) they have a vertex in common, or (2) if G contains an edge between the two subsets. A family of any number of mutually touching, connected vertex sets in G is a *bramble*. A subset of V *covers a bramble* $\mathcal{B}$ if the intersection of the subset with every element of $\mathcal{B}$ is not empty, respectively. The *order of a bramble* $\mathcal{B}$ is the size of the smallest subset of G that covers $\mathcal{B}$.

For example, consider the grid $G_{17,17}$. Let *UPPER* be the set of the seventeen uppermost vertices of $G_{17,17}$, and let *LEFT* be the set of the seventeen leftmost vertices

of $G_{17,17}$. Since *UPPER* and *LEFT* have a vertex in common – namely, the upper leftmost vertex in the graph – *UPPER* and *LEFT* touch. Therefore by definition, the set $UPPERLEFT = \{UPPER, LEFT\}$ is a bramble. Define the set *upperleftvertex* $= \{v : v$ is the upper leftmost vertex of $G_{17,17}\}$. Then *upperleftvertex* covers *UPPERLEFT*, since *upperleftvertex* is a subset of the vertices of $G_{17,17}$, $upperleftvertex \cap UPPER \neq \emptyset$, and $upperleftvertex \cap LEFT \neq \emptyset$. As $|upperleftvertex| = 1$, there cannot be a smaller subset of $G_{17,17}$ that covers *UPPERLEFT* (the only smaller set is $\emptyset$, whose intersection with *UPPER* and *LEFT* is necessarily empty, and so cannot cover *UPPERLEFT*). Therefore, the order of *UPPERLEFT* is 1.

Finally, we have the following theorem:

Theorem 4 ([38])**.** *Let $k \geq 0$ be an integer. A graph G has treewidth $\geq k$ if and only if it contains a bramble of order $> k$.*

2.2 A CSP view of Shape-Free Grid Coloring

In this section, we formalize the notion of Shape-Free Grid Coloring in terms of computational problems, ensure that our definition is robust, and translate the resulting problem family into the language of CSPs in an effort to classify its computational complexity.

2.2.1 A formal notion of Grid Coloring

We begin with a relatively concrete example derived from the 17x17 4-Color Problem – Monochromatic-Rectangle-Free Grid Coloring (MONO-RECT). Define MONO-RECT as the following language in $\{0,1\}^*$:

MONO-RECT $= \{n, m, c : G_{n,m}$ is c-colorable such that no monochromatic rectangles exist$\}$,

where a *rectangle* is a set of four vertices arranged in a rectangular shape, i.e. $\{(i,j),(i+a,j),(i,j+b),(i+a,j+b)\}$ for integers a, b where

- $-i \leq a < n - i$,
- $-j \leq b < m - j$,
- $a \neq 0$, and
- $b \neq 0$.

and a *monochromatic rectangle* is a rectangle whose vertices are all assigned the same color.

However, we immediately arrive at following observation:

Theorem 5. *Either there exists a polynomial-time algorithm to decide* MONO-RECT *that does not explicitly construct c-colorings of* $G_{n,m}$*, or* MONO-RECT $\notin$ NP.

The proof will proceed as follows. We show that if certificates of MONO-RECT are in fact the natural certificate (that is, c-colorings of $G_{n,m}$), then the certificate size of MONO-RECT is exponentially large in the input size, and therefore MONO-RECT cannot be contained in NP.

Lemma 5.1. *Encodings of c-colorings of* $G_{n,m}$ *are exponentially large in the input size of* MONO-RECT.

Proof. An input encoding of an instance of MONO-RECT consists of the *values* of n, m, and c, which requires $\Theta(\log(n) + \log(m) + \log(c))$ bits. WLOG, encodings of c-colorings of $G_{n,m}$ require, at a minimum, a value in $[0, c-1]$ for each of the $n \times m$

cells of $G_{n,m}$ representing the color assigned to each respective cell, which gives an encoding size of c-colorings of $G_{n,m}$ of $\Omega(\log(c) \cdot n \cdot m)$.

However, $(\log(c) \cdot n \cdot m)$ is at least exponential in $(\log(n) + \log(m) + \log(c))$, and the lemma follows. ■

Now we can immediately prove Theorem 5.

Proof of Theorem 5. One of the following two possibilities must be the case: either (1) there exists a non-constructive algorithm to decide MONO-RECT (*non-constructive* in the sense that the algorithm evaluates some function of the input (n, m, c) but does not actually construct a c-coloring of $G_{n,m}$), or (2) all algorithms to decide MONO-RECT must construct c-colorings of $G_{n,m}$.

Suppose the latter. Then the certificates of MONO-RECT instances are c-colorings of $G_{n,m}$ for some fixed n and m. However, by Lemma 5.1, the size of these certificates is exponential in the size of MONO-RECT's instance input size. But by definition of NP, problems contained in NP must have polynomially bounded certificate size, so assuming the supposition, MONO-RECT $\notin$ NP.

As a result, either there must exist some "fast" (i.e., polynomial-time), non-constructive technique to decide instances of MONO-RECT by evaluating a function of n, m and c without actually constructing a legal coloring, or any verifier must spend exponential time in the input to examine a candidate-coloring, and the theorem follows. ■

However (perhaps in part for aesthetic reasons), we find this situation unsatisfactory. That is, we would really prefer a problem that is a natural candidate for either being in P or being NP-complete. If we want to get a problem with purely numerical inputs inside NP, we can use a standard padding trick in complexity theory of

representing the same problem with a *unary* encoding. Define the following modified version of MONO-RECT as a unary language:

U-MONO-RECT $= \{n, m, c : G_{n,m}$ is c-colorable such that no monochromatic rectangles exist$\}$,

where the input string (n, m, c) is given in unary, and the remainder of the problem is identical. However, we then get the following:

Theorem 6. U-MONO-RECT $\in$ SPARSE.

We cite the standard definition of the complexity class SPARSE from the Complexity Zoo[1]: "The class of decision problems for which `YES` instances of size n is upper-bounded by a polynomial in n."

Proof of Theorem 6. All unary languages are necessarily sparse languages, since for each input length n, a unary language contains at most one value of length n and at most n values of length at most n. By the definition of SPARSE, the theorem follows. ■

However, by applying the following theorem of Mahaney,

Theorem 7 ([31])**.** *If* SPARSE *intersects* NPC*, then* P = NP*.*

we can draw the following corollary:

Corollary. *Assume* P $\neq$ NP*. Then by Theorems 6 and 7,* U-MONO-RECT $\notin$ NPC*.*

2.2.2 Avoiding the class SPARSE

To have a truly robust problem definition (in other words, one that allows us to decide whether "Grid Coloring" is "easy" or "hard" in the usual sense), we would prefer to find an appropriate definition that allows us to classify the problem in terms of either P or NP-hard.

Therefore, we will apply the following natural modification to U-MONO-RECT:

RECT-EXTEND = $\{n, m, c, f : f$ is a partial, proper c-coloring of $G_{n,m}$ that can be extended to a total, proper c-coloring of $G_{n,m}\}$,

where the first part of the input string (n, m, c) is given in *binary*[2] and the *partial coloring function*, f, is specified in binary as well. f will be given in the input as a series of nm values each represented by *precisely* $\log(c) + 1$ bits. That is, we let f be a string of nm values, $c_{x,y}$, for $0 \leq c_{x,y} \leq c$ given in *row-major order* of the vertices (x, y) with the beginning of each value padded with zeroes when required to force it to be $\log(c) + 1$ bits. The (implicit) values x and y will jointly uniquely specify a vertex in $G_{n,m}$, and the value $c_{x,y}$ will specify the fixed, initial color assigned to vertex (x, y). Further, a *proper c-coloring* is any assignment of c colors to vertices in $G_{n,m}$ such that no monochromatic rectangles exist. A *partial, proper c-coloring* is a proper c-coloring that does not assign a color to at least one vertex of $G_{n,m}$, with potentially as few as no such assignments. A *total, proper c-coloring* is a proper c-coloring that assigns a color to every vertex in the grid graph.

In particular, we highlight that in order to allow the encoding scheme the satisfy

[2]There is no reason why (n, m, c) could not remain encoded in unary, since as will be shown, f will always dominate the input size of RECT-EXTEND. However, as binary encodings are generally far more common and as we do not need to retain a unary encoding to "push" RECT-EXTEND inside NP, we revert to binary for simplicity's sake.

the condition of a *partial, proper c-coloring*, the range of allowed values allowed for the $c_{x,y}$ is $[0, c]$, which is $c + 1$ distinct values, requiring $\log(c) + 1$ bits to represent. The first c of these values specify the c different colors allowed, whereas the values $c_{x,y} \geq c$ is interpreted as "no coloring for vertex (x, y)." Therefore, WLOG, we can read the requirement that f be a *partial*, proper c-coloring as a requirement that *at least one* of the $c_{x,y}$ be given the value c.

The definition of RECT-EXTEND (and in particular, the consequentially exponential blow-up in input size) allows us to "bypass" the class SPARSE and any restricting consequences of Theorem 7:

Theorem 8. RECT-EXTEND $\notin$ SPARSE.

The proof will proceed as follows: The goal is to identify a natural subset of RECT-EXTEND instances that are both guaranteed to be `YES` instances *and* that is exponential in the size of the input N, for infinitely many N. This, in turn, will guarantee that there can be no polynomial in the input size N that bounds the number of `YES` instances of RECT-EXTEND as N grows, leading to Theorem 8.

Proof of Theorem 8. Consider the set of all grids $G_{2,m}$. That is, let the dimension n=2. Additionally, let the number of allowed colors, c, be 2, and let the partial coloring function f be chosen later. Consider the situations that arise as m increases to infinity.

In particular, we have grids of two columns (resp. rows), and for any fixed m, the set of legal colorings include those where the first column is entirely colored by the first color and the second column is entirely colored by the second color. (In fact, any coloring where one can scan each column and only encounter two of the same color in at most one of the two columns is legal, but we only need the above subset.)

Then the input size is given by:

$$\begin{aligned} N &= 2 + \log(m) + 2 + (2m(1+1)) \\ &= \Theta(m) \end{aligned}$$

where in the first expression, the first term is the number of bits required to represent an input of "$n = 2$," the second term is the number of bits required to represent an input of "m," the third term is the number of bits required to represent "$c = 2$," and the final term is the number of bits required to represent the partial coloring function f (i.e., $nm(\log(c) + 1)$ for $n = 2, c = 2$).

Finally, we allow f to arbitrarily either select (or not select) up to $2m - 1$ cells in $G_{2,m}$, and if a cell in the first column is selected, f gives that cell the first color (WLOG, "0"); if a cell in the second column is selected, f gives that cell the second color (WLOG, "1"); and if a cell is not selected, f gives that cell no color (WLOG, "2"). WLOG, assume the vertex $(0, 0)$ is always not selected (in order to fit the requirement that at least one cell is always uncolored), then there are $2^{2m-1} = \Theta(4^m)$ such ways to choose whether to select or not select the remaining $2m - 1$ cells. By construction, each of these choices correspond to a `YES` instance of RECT-EXTEND. But since $N = \Theta(m)$, no polynomial in N bounds the number of `YES` instances of RECT-EXTEND, and the theorem follows. ■

2.2.3 Shape-Free Grid Coloring as a CSP

With a computational language in hand that is representative of Monochromatic-Rectangle-Free Grid Coloring, we now generalize this definition to the family of problems on grid graphs with arbitrary constraints (i.e., we will only consider the structural properties of the problem). The property in common between all such problems is the underlying grid graph structure, which we will now view in terms of the set of constraints between different vertices.

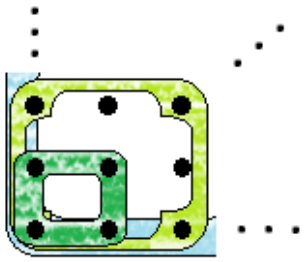

Figure 2.1: Overlapping hyperedges of an example hypergraph of $\mathcal{G}$

We begin by defining the family of graphs $\texttt{Fam}(G_{n,m}) := \bigcup_{n\geq 2, m\geq 2}\{G_{n,m}\}$. We then (implicitly) construct a family of hypergraphs, $\mathcal{G}$, by a bijection from $\texttt{Fam}(G_{n,m})$. In particular, for every $G_{n,m} \in \texttt{Fam}(G_{n,m})$, let there be a matching hypergraph $\mathcal{G}_{n,m} \in \mathcal{G}$ such that $V(\mathcal{G}_{n,m}) = V(G_{n,m})$ and $E(\mathcal{G}_{n,m}) = \{e : e \text{ is a rectangle in } G_{n,m}\}$. Some hyperedges of an example hypergraph of $\mathcal{G}$ are shown in Fig. 2.1.

Now we show the following:

Theorem 9. *$\mathcal{G}$ has unbounded treewidth.*

The proof begins with the following lemma:

Lemma 9.1. *Fix some n and m. Then $G_{n,m}$ contains a bramble of order* $\min(n, m)$.

Proof. Let an *i-j-cross of $G_{n,m}$* be the union of the vertices in the ith row and jth column of the graph. Let $\mathcal{B}^{\times}$ be the union of all such i-j-crosses. Then by definintion, $\mathcal{B}^{\times}$ is a bramble. Observe that any given row or column in $G_{n,m}$ covers $\mathcal{B}^{\times}$. The lowest number of vertices across all row and columns is $\min(n, m)$. However, if any less vertices are chosen for the cover than those in an entire row or column, at least one row or column in an i-j-cross of $\mathcal{B}^{\times}$ will not be covered. Thus, to cover every

cross in $\mathcal{B}^{\times}$, at least $\min(n, m)$ vertices must be used. ■

Lemma 9.2. *Fix some n and m. Then the treewidth of $G_{n,m}$ is $\Omega(\min(n, m) - 1)$.*

Proof. By Theorem 4 and Lemma 9.1, the lemma follows. In particular, k of Theorem 4 is $\min(n, m)$ by Lemma 9.1, which implies $\text{tw}(G_{n,m}) \geq \min(n, m) - 1$. ■

And we have the straightfoward consequence:

Corollary. *The treewidth of* $\texttt{Fam}(G_{n,m})$ *is unbounded.*

Now we can prove Theorem 9 by drawing on a result of [3], which states:

Theorem 10 ([3])**.** *The treewidth of a hypergraph H equals the maximum treewidth of its connected components.*

Remark. The definition of a *path from v_1 to v_n* in a hypergraph H is a sequence of vertices $(v_1, ..., v_n)$ such that any two consecutive vertices are contained in a common hyperedge of $E(H)$. We say two vertices are *connected* if there is a path between them. A *connected component* of a hypergraph H is a subgraph $H' = (V', E')$ for $V' \subseteq V(H)$ and arbitrary E' *so long as* any pair of vertices $(u, v) \in H'$ are connected in H' by a path p *only if* $(u, v) \in H$ are connected by p as well.

Proof of Theorem 9. Observe that for each $\mathcal{G}_{n,m} \in \mathcal{G}$, $\mathcal{G}_{n,m}$ has $G_{n,m}$ as a connected component. Each such connected component, for some fixed n and m, has treewidth $\Omega(\min(n, m) - 1)$ by Lemma 9.2. Then by Theorem 10, $\mathcal{G}_{n,m}$ has treewidth at least $\Omega(\min(n, m) - 1)$ as well. Then, in the same vein as the above corollary, $\mathcal{G}$

has unbounded treewidth. ■

Now we define $\mathsf{CSP}(\mathcal{G})$ per the definitions at the end of Section 2.1.1. Finally, we prove the primary (unconditional) theorem of the CSP segment of the thesis:

Theorem 11. *Assume the ETH. Then* $\mathsf{CSP}(\mathcal{G}) \notin$ **PTIME**.

Proof of Theorem 11. First, observe that each hyperedge in every $\mathcal{G}_{n,m} \in \mathcal{G}$ contains precisely four vertices. Clearly $\mathcal{G}$ has bounded hyperedge size. Then we apply Theorem 3 and Theorem 9, and the theorem follows. ■

2.2.4 The NAE is Robust for Grids Assumption

Now we formally introduce the *NAE is Robust for Grids Assumption* in order to relate the computational complexities of $\mathsf{CSP}(\mathcal{G})$ and RECT-EXTEND.

NAE is Robust for Grids Assumption. *Assume* $\mathsf{CSP}(\mathcal{G}) \leq_p$ RECT-EXTEND.

We see this assumption as plausible, *prima facie*, due to its similarity to the SAT $\leq_P$ NAE-SAT NP-completeness reduction. That is, we begin with an arbitrary relation we want to satisfy, and we transform it into a relation where not every variable can take the same value. Allow us muse momentarily about the differences between the two in order to illuminate both why we have had difficulty proving this claim and, consequently, why it could be true or false.

Discussion. One of the least understood and crucial differences between computational problems known to be in P and those known to be NP-complete is in the different types of *implicative, logic statements* they allow. By way of simple example,

consider the different types of implications one can generate in 2-SAT and in 3-SAT. In a typical 2-SAT instance, we are given a set of variables grouped in disjunctive clauses of size 2, e.g. $(x_2 \vee x_3)$. Suppose we assign the variable x_2 the value *false*. Then, for the clause to evaluate to *true*, it must be the case that x_3 is assigned the value *true*. Abstracting this principle, we can rewrite any clause of the form $(x_i \vee x_j)$ as $(\bar{x}_i \rightarrow x_j)$; that is, we get implicative statements like "If x_i is *false*, then x_j is *true*." However, this implies a simple polynomial-time algorithm to decide 2-SAT: Independently attempt to assign *true* to each variable $x_i, \forall i$, then follow the chain of implications. If $x_i \rightarrow \bar{x}_i$ and $\bar{x}_i \rightarrow x_i$ for any i, then the formula is unsatisfiable; otherwise, it is satisfiable. Finally, we observe that the algorithm takes polynomial-time as a result of the simple path-like structure of implications derived from such clauses. (Proof omitted, though examples are abundant in the literature.)

However, in the 3-SAT case, the situation is different. The clauses are now of the form $(x_i \vee x_j \vee x_k)$, which results in the following form of implication: $(x_i \rightarrow (x_j \vee x_k))$. Indeed, by applying the same algorithmic logic as in 2-SAT, we see that the resulting structure of implications forms a *binary tree*, and in order to rule out the possibility that $x_i \rightarrow \bar{x}_i$ (and vice versa), we must "visit" every leaf of this tree. It is a simple matter to see that the size of this tree is exponential in the size of the corresponding 3-SAT formula, and we can conclude that *if* we use the generalization of the 2-SAT algorithm for 3-SAT, *then* we must expend time exponential in the input size in order to arrive at a decision. And of course, the question of whether there exists an algorithm that can perform asymptotically better is equivalent to the Exponential Time Hypothesis, and the question of whether there exists an algorithm that can complete the entire task in polynomial time is equivalent to the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ question.

Further, one can take the view that the theory of NP-completeness is a matter of precisely *embedding* this implicative, tree-like structure in other computational problems. This pattern is perhaps most obvious in graph-theoretic computation problems.

We could ask, "*Why* is 2-COLORABILITY in P and 3-COLORABILITY NP-complete?" A succinct answer in turn could be that 2-COLORABILITY fundamentally demonstrates an implicative, path-like structure, whereas 3-COLORABILITY demonstrates an implicative, tree-like structure.

Applying this same reasoning to problems in Shape-Free Grid Coloring, specifically RECT-EXTEND, and contrasting it with the situation for 3-SAT tells us much. As the size of the input of an instance of 3-SAT grows, we add clauses to the formula (and potentially increase the number of variables as well), and in particular, each clause can be introduced essentially *independently* of the clauses previously found in the formula. However with RECT-EXTEND, this is not the case. As the size of the input of an instance of RECT-EXTEND grows, we add rows (resp. columns) to the grid graph that must be properly colored. Each of the grid cells thus introduced *must* be constrained in a very specific manner.

For example, intially consider an instance with a 2-by-2 grid graph and 2 allowed colors. There are four grid cells, which we will (under some arbitrary ordering) associate with the Boolean variables x_0, x_1, x_2, and x_3. The natural interpretation in terms of a Satisfiability problem would be as NAE-SAT, and the single, corresponding clause is therefore (x_0, x_1, x_2, x_3). Now consider the situation when we increase the value of n from 2 to 3. Now we have a 3-by-2 grid graph and 2 allowed colors, and we will associate the two new grid cells with the Boolean variables x_4 and x_5. The resulting NAE-SAT interpretation would be three clauses: $\{(x_0, x_1, x_2, x_3), (x_0, x_1, x_4, x_5), (x_2, x_3, x_4, x_5)\}$. It is this apparent *lack of independence* in introducing additional clauses and variables – the underlying, implicative, grid-like structure – that is the key, outstanding difficulty in relating Shape-Free Grid Coloring problems to known NP-complete problems.

One possibility, of course, is that the introduction of the partial coloring function for RECT-EXTEND allows an NP-completeness reduction by "breaking" the inherent

dependence between the coloring of grid cells. An alternate, smaller step would be to definitively show that $\mathsf{CSP}(\mathcal{G})$ reduces to RECT-EXTEND, as the former of which is intentionally designed to ignore every aspect of the underlying computational problem *except* its grid-like structure. We take the view that our work up to this point is in pursuit of the second, hopefully simpler goal.

We use the NAE is Robust for Grids Assumption for the following theorem:

Theorem 12. *Assume the ETH and NAE is Robust for Grids Assumption. Then* RECT-EXTEND $\notin$ **PTIME.**

Proof. Suppose not. Then from Theorem 11, assuming the ETH, we have that $\mathsf{CSP}(\mathcal{G})$ is not solvable in polynomial time in the worst case. Assuming the NAE is Robust for Grids Assumption, we can transform any instance of $\mathsf{CSP}(\mathcal{G})$ into an instance of RECT-EXTEND in polynomial time. Then, by supposition, we can solve RECT-EXTEND in polynomial time. This allows us to extract a `YES` or `NO` decision for $\mathsf{CSP}(\mathcal{G})$ in polynomial time, which is a contradiction, and the theorem follows. ■

Finally, we point out that Theorem 12 does *not* automatically extend to a corollary that RECT-EXTEND is NP-complete, even if we could remove its reliance on the NAE is Robust for Grids Assumption. That is, due to [10], it is known that there is no dichotomy between P and NP-complete for CSPs of $\mathsf{CSP}(\mathcal{G})$'s form (for more information, also see [7]).

2.3 Lower Bounds in Communication Complexity

In this section, we demonstrate that Monochromatic-Rectangle-Free Grid Coloring provides lower bounds for the communication complexity of a group of similar pred-

icates in the multi-party communcation complexity model. We begin by discussing the notion of multi-party communication complexity, broadly defined. We then give three explicit, different examples of such predicates and prove a lower bound for each. We end by making explicit a possible, broader connection between Ramsey-theoretic mathematics and lower bounds in communication complexity.

2.3.1 The Multi-Party Communication Complexity Model

In the *Multi-Party Communication Complexity Model*, a generalization of the earlier NOF model, there are $k \geq 3$ parties $P_0, ..., P_{k-1}$ with unbounded computational power that want to exchange information in order to compute a 0-1 predicate of k integers, $A = \{a_0, ..., a_{k-1}\}$ where $a_i \in [0, n]$, $\forall i$, whose input has been divided according to $\Phi = \{\phi_0, ..., \phi_{k-1}\}$, a k-partition of A. The ith party, P_i, receives the value of every a_j except for those in ϕ_i. Note that in the case $\phi_i = \{a_i\}$, $\forall i$, that this is precisely the NOF model. In general however, per the typical definition of a k-partition, the sets $\phi_i \subsetneq A$ can be allowed to consist of any subset of A so long as each a_j is assigned to at least one ϕ_i and $\phi_i \neq A, \forall i$.

We let the parties P_i exchange information by broadcasting bits according to a round-robin ordering, $P_0, P_1, ...$, across a shared communication channel that all parties can freely monitor. The definitions of a protocol, a broadcast history, a situation, S_n, H_n, validity of a protocol follow from the NOF model without change. For clarity's sake, we make a small modification to the definition of the communication complexity of a predicate by saying that the multiparty communication complexity of f, where f is the predicate in question, *with respect to the fixed partition* Φ, is the minimum number of bits broadcast by any protocol that correctly compute f across all possible protocols as the range of allowed values, n, grows toward infinity. Further, we will refer to the *private inputs of party* P_i by $\zeta(P_i) = A \backslash \phi_i$.

We will now restrict our study, for simplicity's sake, to the case when the number

of parties and inputs k is even and the partitions of A symmetrically cover half of A. Specifically, $\phi_i = \{a_{i \mod k}, a_{(i+1) \mod k}, ..., a_{(i+\frac{k}{2}-1) \mod k}\}$. For example, let $k = 4$, then $A = \{a_0, a_1, a_2, a_3\}$, and then $\phi_0 = \{a_0, a_1\}$, $\phi_1 = \{a_1, a_2\}$, $\phi_2 = \{a_2, a_3\}$, $\phi_3 = \{a_3, a_0\}$, and then $\zeta(P_0) = \{a_2, a_3\}$, $\zeta(P_1) = \{a_3, a_0\}$, $\zeta(P_2) = \{a_0, a_1\}$, and $\zeta(P_3) = \{a_1, a_2\}$.

2.3.2 The Lower Bound

We begin with a statement of the three example predicates:

1. Let the predicate *Exactly-*$\frac{k}{2}n$ be true for k integers $a_0, ..., a_{k-1}$ if and only if $a_0 + \cdots + a_{k-1} = \frac{k}{2}n$. That is, we define $S_n(\textit{Exactly-}\frac{k}{2}n) := \{\bar{v} \in H_n : \sum_i v_i = \frac{k}{2}n\}$.

2. Let the predicate *PlanarExactly-n* be true for k integers $a_0, ..., a_{k-1}$ if and only if

$$\begin{aligned} & a_0 + a_{\frac{k}{2}} = n \\ \wedge\ & a_1 + a_{\frac{k}{2}+1} = n \\ & \cdots \\ \wedge\ & a_{\frac{k}{2}-1} + a_{k-1} = n. \end{aligned}$$

   That is, we define $S_n(\textit{PlanarExactly-n}) := \{\bar{v} \in H_n : \bigwedge_{i=0}^{\frac{k}{2}-1}(v_i + v_{i+\frac{k}{2}} = n)\}$.

3. Let the predicate *LinearExactly-n* be true for k integers $a_0, ..., a_{k-1}$ if and only

if

$$a_0 + a_{\frac{k}{2}} = n$$
$$\vee\ a_1 + a_{\frac{k}{2}+1} = n$$
$$\ldots$$
$$\vee\ a_{\frac{k}{2}-1} + a_{k-1} = n.$$

That is, we define $S_n(\textit{LinearExactly-}n) := \{\bar{v} \in H_n : \bigvee_{i=0}^{\frac{k}{2}-1}(v_i + v_{i+\frac{k}{2}} = n)\}$.

Remark. The choice of name for the latter two predicates are not intended to imply a strict geometric interpretation for their respective solution spaces, S_n. While, as will be shown below in further detail, for $k = 4$ $S_n(\textit{PlanarExactly-}n)$ is a square-shaped plane embedded in the four-dimensional hypercube of situations, for $k \geq 6$ the geometric structure of $S_n(\textit{PlanarExactly-}n)$ is *not* a plane. And in fact, $S_n(\textit{LinearExactly-}n)$ is never a *line*, but instead takes its name from the linear constraints it imposes on (at least one of) the parties' inputs.

Now we prove two lemmas about the relationships between the solution spaces, S_n, of these predicates that we will need later.

Lemma 13.1. $S_n(\textit{PlanarExactly-}n) \subsetneq S_n(\textit{Exactly-}\frac{k}{2}n)$.

Proof. First we show containment, then inequality by counterexample.

$S_n(\textit{PlanarExactly-}n) \subseteq S_n(\textit{Exactly-}\frac{k}{2}n)$: Fix any $\bar{v} \in S_n(\textit{PlanarExactly-}n)$. Since $\bigwedge_{i=0}^{\frac{k}{2}-1}(v_i + v_{i+\frac{k}{2}} = n)$, then $\sum_i v_i = \frac{k}{2}n$, because there are $\frac{k}{2}$ pairs of coordinates that each sum to n, which jointly equal $\frac{k}{2}n$.

$S_n(\textit{PlanarExactly-}n) \neq S_n(\textit{Exactly-}\frac{k}{2}n)$: Let $k = 4, n = 4$, and $\bar{v} = \langle 1, 3, 1, 3\rangle \in H_n$. Then $\sum_i v_i = 1+3+1+3 = 8 = \frac{4}{2}4 = \frac{k}{2}n$, so $\bar{v} \in S_n(\textit{Exactly-}\frac{k}{2}n)$. But let $i = 0$,

then $v_i + v_{i+\frac{k}{2}} = v_0 + v_2 = 1 + 1 = 2 \neq 4 = n$, so $\bar{v} \notin S_n(\textit{PlanarExactly-}n)$. ■

Lemma 13.2. $S_n(\textit{PlanarExactly-}n) \subsetneq S_n(\textit{LinearExactly-}n)$.

Proof. First we show containment, then inequality by counterexample.

$S_n(\textit{PlanarExactly-}n) \subseteq S_n(\textit{LinearExactly-}n)$: Fix any $\bar{v} \in S_n(\textit{PlanarExactly-}n)$. Since $\bigwedge_{i=0}^{\frac{k}{2}-1}(v_i + v_{i+\frac{k}{2}} = n)$ – that is, all of the pairs of coordinates, $(v_i, v_{i+\frac{k}{2}})$, sum to n – then it is also the case that at least one of the pairs of coordinates sum to n. That is, $\bigvee_{i=0}^{\frac{k}{2}-1}(v_i + v_{i+\frac{k}{2}} = n)$; just pick any pair of coordinates $(v_i, v_{i+\frac{k}{2}})$.

$S_n(\textit{PlanarExactly-}n) \neq S_n(\textit{LinearExactly-}n)$: Let $k = 4, n = 4$, and $\bar{v} = \langle 1, 0, 3, 0\rangle \in H_n$. Let $i = 0$, then $v_i + v_{i+\frac{k}{2}} = v_0 + v_2 = 1 + 3 = 4 = n$, so $\bar{v} \in S_n(\textit{LinearExactly-}n)$. But let $i = 1$, then $v_i + v_{i+\frac{k}{2}} = v_1 + v_3 = 0 + 0 \neq 4 = n$, so $\bar{v} \notin S_n(\textit{PlanarExactly-}n)$. ■

A large portion of the following proof will closely parallel the communication complexity-theoretic lower bound proof found in Chapter 1 for *Exactly-n* in the NOF model. We begin with the following crucial definitional extension:

Define a *forbidden k-rectangle* (resp. a *forbidden k-hyperrectangle*), for even k, to be a set of k distinct points $\bar{v}_0, ..., \bar{v}_{k-1} \in H_n$ if there is a point $\bar{w} \in H_n$ such that, for each i, $\bar{w}$ differs from $\bar{v}_i$ only in coordinates $i \mod k, (i+1) \mod k, ..., (i + \frac{k}{2} - 1) \mod k$.

Further, we define the *view of P_i at $\bar{v}_j$ at time t*, written $\texttt{view}(\zeta(P_i), b)_{\bar{v}_j, t}$, as the total sum of information available to P_i given the parties' global inputs (conditioned on $\bar{v}_j$), P_i's private inputs (conditioned on $\bar{v}_j$), and the current broadcast history (resulting from the application of some protocol $\mathcal{P}$ up to time $t-1$), which it can use to decide which bit to broadcast at time t.

Finally, we will primarily make use of the predicate *PlanarExactly-n* for what follows, using Lemmas 13.1 and 13.2 to connect the argument to the predicates *Exactly-*$\frac{k}{2}n$ and *LinearExactly-n* at the end.

Now we prove the following:

Theorem 13. *Let* $S_n = \{\bar{v} \in H_n : \bigwedge_{i=0}^{\frac{k}{2}-1}(v_i + v_{i+\frac{k}{2}} = n)\}$. *The communication complexity of any k-party protocol for PlanarExactly-n is bounded below by the logarithm of the minimum number of colors required to color the points of* S_n *so that no forbidden k-rectangles are colored monochromatically.*

First, we need the following lemma:

Lemma 13.3. *Let* $\mathcal{P}$ *be a protocol for parties* $P_0, ..., P_{k-1}$. *Let* $H_n = \{0, ..., n\}^k$ *be the hypercube of situations, and let* $\bar{w} = \langle w_0, ..., w_{k-1}\rangle$ *be any point in* H_n. *If* $\bar{v}_0, ..., \bar{v}_{k-1}$ *is a forbidden k-rectangle for* $\bar{w}$ *and the complete broadcast histories at the* $\bar{v}_j$ *are identical, that is*

$$\alpha = \mathcal{P}(\bar{v}_0) = \mathcal{P}(\bar{v}_1) = \cdots = \mathcal{P}(\bar{v}_{k-1}), \tag{2.2}$$

then $\mathcal{P}(\bar{w}) = \alpha$.

Proof. We prove the lemma by induction on the length of the broadcast history, that for all i, the i^{th} bit broadcast at $\bar{w}$ is the same as the i^{th} bit broadcast at $\bar{v}_{i(\mathrm{mod}\ k)}$.

Base step: If $|\alpha| = 0$, then every process immediately halts on $\bar{w}$ since it immediately halts on $\bar{v}_i$. That is, since $\bar{v}_0, ..., \bar{v}_{k-1}$ is a forbidden k-rectangle for $\bar{w}$, $\zeta(P_i)_{\bar{w}} = \zeta(P_i)_{\bar{v}_i}, \forall i$, by the definition of a forbidden k-rectangle, because $A_{\bar{w}}$ only differs from $A_{\bar{v}_i}$ in the a_j that are not in $\zeta(P_i)$ for each situation, $\forall i$. Further, $b = \epsilon$ for each situation, since $|\alpha| = 0$. Therefore, $\texttt{view}(\zeta(P_i), b)_{\bar{v}_i,t}$ are equal $\forall i$, and the

parties must act identically for any fixed, deterministic protocol $\mathcal{P}$.

Inductive step: Assume that for all histories of length $< t$, the inductive hypothesis is true. Consider the t^{th} bit to be broadcast. By induction, $P_{t(\mathrm{mod}\ k)}$ sees the same broadcast history b at $\bar{w}$ and at $\bar{v}_{t(\mathrm{mod}\ k)}$. As before, by the definition of a forbidden k-rectangle, $\zeta(P_i)_{\bar{w}} = \zeta(P_i)_{\bar{v}_i}, \forall i$, so the $\texttt{view}(\zeta(P_i), b)_{\bar{v}_i,t} = \texttt{view}(\zeta(P_i), b)_{\bar{w},t}$ $\forall i$. Therefore, $P_{t(\mathrm{mod}\ k)}$ broadcasts the same bit at time t at $\bar{w}$ as it would at time t at $\bar{v}_{t(\mathrm{mod}\ k)}$. ■

Note that Lemma 13.3 holds in the case of invalid protocols as well as valid ones and is *independent of the predicate in question.* For clarity's sake, we explicitly construct an example of the above lemma. Let $n = 1$, let $k = 4$ and let $\bar{w} = \langle 1, 1, 1, 1\rangle$. Let the following points form a forbidden 4-rectangle for $\bar{w}$: $\bar{v}_0 = \langle 0, 0, 1, 1\rangle, \bar{v}_1 = \langle 1, 0, 0, 1\rangle, \bar{v}_2 = \langle 1, 1, 0, 0\rangle$, and $\bar{v}_3 = \langle 0, 1, 1, 0\rangle$. In particular, note that $\zeta(P_i)$ at $\bar{v}_i$, for all i, is $\{a_{(i+2)\ \mathrm{mod}\ k}, a_{(i+3)\ \mathrm{mod}\ k}\} = \{1, 1\}$ respectively. By hypothesis, the broadcast histories b at all the $\bar{v}_i$ are identical as well. As a result, the views of each party P_i at each $\bar{v}_i$ at times t, that is $\texttt{view}(\zeta(P_i), b)_{\bar{v}_i,t} = \texttt{view}(\zeta(P_i), b)_{\bar{w},t}$, $\forall i, t$. Thus, assuming the hypothesis and for any fixed, deterministic protocol $\mathcal{P}$, the parties will send the same bits at points in the forbidden 4-rectangle as they do at $\bar{w}$.

Note that this example extends to sets of points of the simple form $\{\langle a, a, b, b\rangle$, $\langle b, a, a, b\rangle$, $\langle b, b, a, a\rangle$, $\langle a, b, b, a\rangle\}$ for $\bar{w} = \langle b, b, b, b\rangle$ and integers $a, b \in [0, n]$ such that $v_i \in S_n$, $\forall i$ (and others, as described below). Further note that points of this form constitute 2-dimensional planes embedded in the 4-dimensional hypercube of situations, H_n. A geometric interpretation of this (simple) example is shown in Fig. 2.2.

Lemma 13.4. *A k-party protocol $\mathcal{P}$ is not valid for the predicate PlanarExactly-n if it assigns the same broadcast history to all points of a forbidden k-rectangle of S_n.*

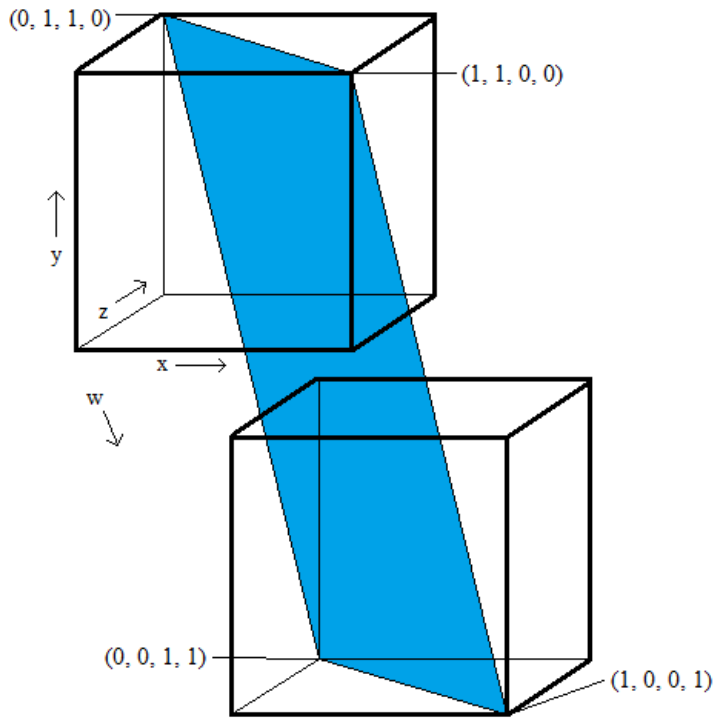

Figure 2.2: A simple forbidden 4-rectangle

Proof. Suppose $\bar{v}_0, ..., \bar{v}_{k-1}$ is a forbidden k-rectangle on S_n for the point $\bar{w}$, such that the equality relationship in Equation (2.1) holds. By geometry, $\bar{w}$ is not on S_n. (In fact, for each of the $\bar{v}_i \in S_n$ that are a forbidden k-rectangle for $\bar{w}$, $\bar{w}$ differs from the $\bar{v}_i$ in precisely one of the coordinates in each of the summations that are a part of the $\bigwedge_{i=0}^{\frac{k}{2}-1}(v_i + v_{i+\frac{k}{2}} = n)$ expression by definition of a forbidden k-rectangle.) By Lemma 13.3, $\mathcal{P}(\bar{w}) = \alpha$. Therefore, $\mathcal{P}$ is not valid for *PlanarExactly-n*. ■

Define the integer $\mu_k(n)$ to be the smallest number of colors required to color the points of S_n so that no forbidden k-rectangle on S_n is colored monochromatically. Now we can prove a general lower bound for *PlanarExactly-n*.

Proof of Theorem 13. Consider a protocol $\mathcal{P}$ that computes *PlanarExactly-n*. Let each distinct broadcast history $\mathcal{P}(\bar{v})$, for $\bar{v} \in H_n$, define a color. (Note that this colors

all the points of S_n.) If some forbidden k-rectangle on S_n is colored monochromatically, then by Lemma 13.4, $\mathcal{P}$ is not valid for *PlanarExactly-n*, which is a contradiction. Therefore, there must be more than $\mu_k(n)$ distinct broadcast histories, and hence in some situation, $\Omega(\log(\mu_k(n)))$ bits must be communicated. ■

Again, for clarity's sake, we explicitly demonstrate a construction of S_n from which $\mu_k(n)$ is directly derived. In particular, we extend the previous geometric illustration of forbidden 4-rectangles to the case $n = 2, k = 4$. Consider the following points in the situation space, H_n, for *PlanarExactly-n*: $\bar{v}_0 = \langle 0, 0, 2, 2\rangle, \bar{v}_1 = \langle 2, 0, 0, 2\rangle, \bar{v}_2 = \langle 2, 2, 0, 0\rangle$, and $\bar{v}_3 = \langle 0, 2, 2, 0\rangle$, which form a forbidden 4-rectangle for the point $\bar{w} = \langle 2, 2, 2, 2\rangle$. Consider the set of lines that connect each of the four points in the forbidden 4-rectangle with minimum Euclidean distance. Observe that the midpoint of each of these lines is in H_n. (In fact, these nine points form a 3-by-3 grid in which we must avoid monochromatic rectangles.)

An embedding of the forbidden 4-rectangles in the 4-dimensional hypercube of situations is shown in Fig 2.3. A geometric interpretation of the resulting set of forbidden 4-rectangles embedded in the 2-dimensional plane is shown in Fig 2.4.

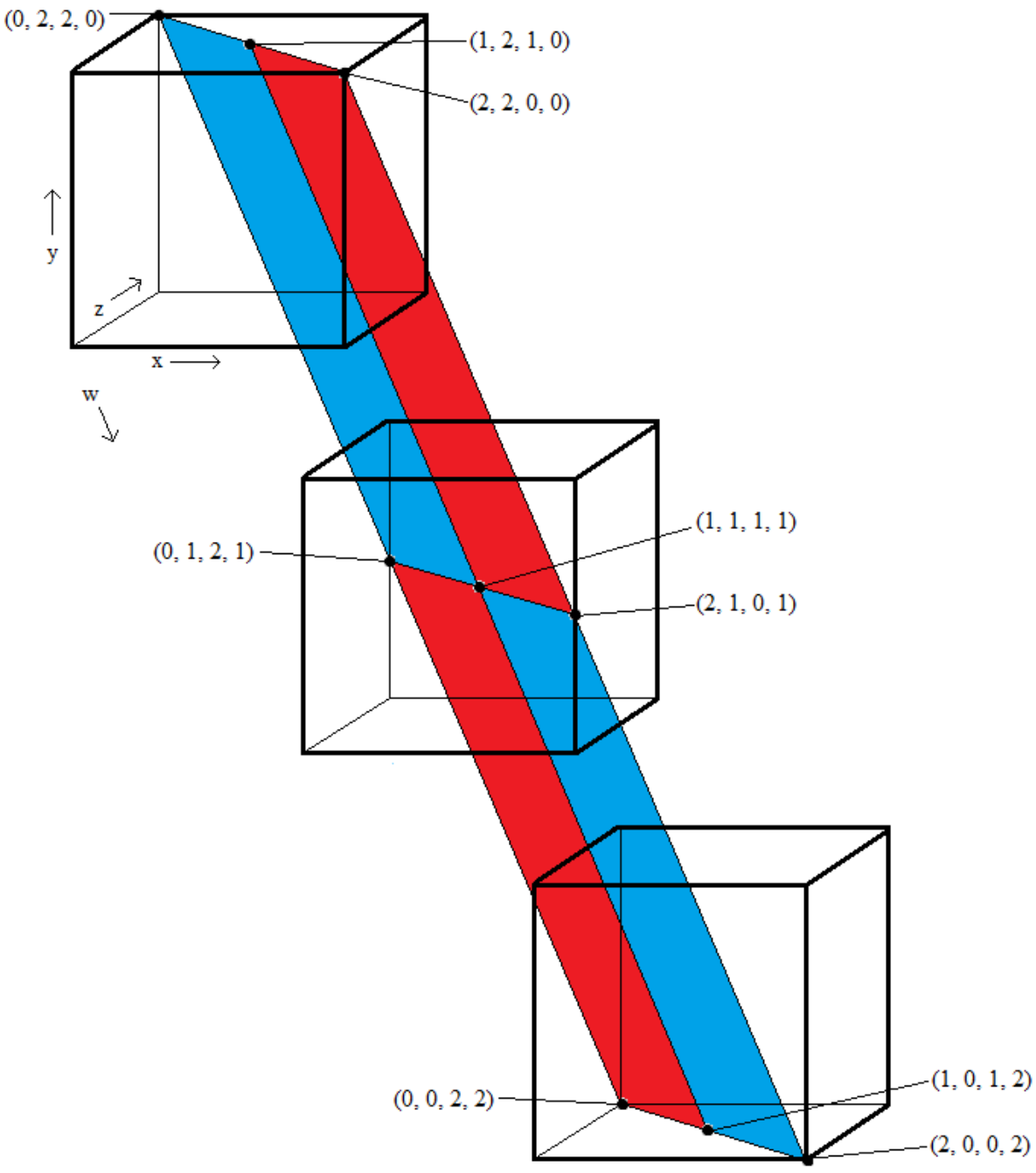

Figure 2.3: 4D-embedding of forbidden k-rectangles for *PlanarExactly-n* (n=2, k=4)

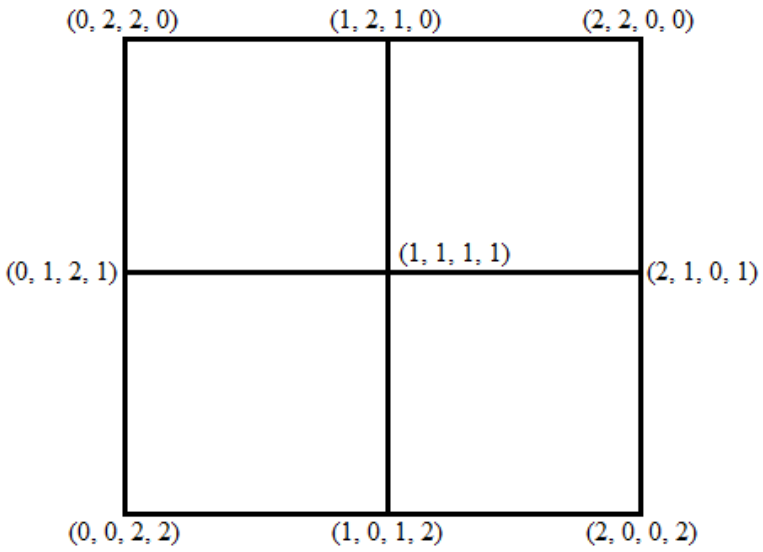

Figure 2.4: 2D-embedding of forbidden k-rectangles for *PlanarExactly-n* (n=2, k=4)

In particular, the group[3] of the nine forbidden 4-rectangles and the points for which they form the rectangle from the above example are given below. Recall $n = 2$ and observe that for each of the $\bar{v}_{i,j}$, $\bigwedge_{K=0}^{\frac{k}{2}-1}(v_{i,j,K} + v_{i,j,(K+\frac{k}{2})} = n)$, whereas $\forall \bar{w}_i, \exists j$ such that $v_j + v_{j+\frac{k}{2}} \neq n$. (That is, $\bar{v}_{i,j} \in S_n$, $\forall i, \forall j$, and $\bar{w}_i \notin S_n$, $\forall i$.)

[3]Note the change in index notation when referring to *groups* of forbidden k-patterns (resp. forbidden k-rectangles), as discussed in the footnote of Section 1.3.1.

$$(\{\bar{v}_{0,0} = \langle 0,1,2,1\rangle, \bar{v}_{0,1} = \langle 1,1,1,1\rangle, \bar{v}_{0,2} = \langle 1,2,1,0\rangle, \bar{v}_{0,3} = \langle 0,2,2,0\rangle\}, \bar{w}_0 = \langle 1,2,2,1\rangle)$$

$$(\{\bar{v}_{1,0} = \langle 0,1,2,1\rangle, \bar{v}_{1,1} = \langle 2,1,0,1\rangle, \bar{v}_{1,2} = \langle 2,2,0,0\rangle, \bar{v}_{1,3} = \langle 0,2,2,0\rangle\}, \bar{w}_1 = \langle 2,2,2,1\rangle)$$

$$(\{\bar{v}_{2,0} = \langle 0,0,2,2\rangle, \bar{v}_{2,1} = \langle 1,0,1,2\rangle, \bar{v}_{2,2} = \langle 1,2,1,0\rangle, \bar{v}_{2,3} = \langle 0,2,2,0\rangle\}, \bar{w}_2 = \langle 1,2,2,2\rangle)$$

$$(\{\bar{v}_{3,0} = \langle 0,0,2,2\rangle, \bar{v}_{3,1} = \langle 2,0,0,2\rangle, \bar{v}_{3,2} = \langle 2,2,0,0\rangle, \bar{v}_{3,3} = \langle 0,2,2,0\rangle\}, \bar{w}_3 = \langle 2,2,2,2\rangle)$$

$$(\{\bar{v}_{4,0} = \langle 1,1,1,1\rangle, \bar{v}_{4,1} = \langle 2,1,0,1\rangle, \bar{v}_{4,2} = \langle 2,2,0,0\rangle, \bar{v}_{4,3} = \langle 1,2,1,0\rangle\}, \bar{w}_4 = \langle 2,2,1,1\rangle)$$

$$(\{\bar{v}_{5,0} = \langle 1,0,1,2\rangle, \bar{v}_{5,1} = \langle 2,0,0,2\rangle, \bar{v}_{5,2} = \langle 2,2,0,0\rangle, \bar{v}_{5,3} = \langle 1,2,1,0\rangle\}, \bar{w}_5 = \langle 2,2,1,2\rangle)$$

$$(\{\bar{v}_{6,0} = \langle 0,0,2,2\rangle, \bar{v}_{6,1} = \langle 1,0,1,2\rangle, \bar{v}_{6,2} = \langle 1,1,1,1\rangle, \bar{v}_{6,3} = \langle 0,1,2,1\rangle\}, \bar{w}_6 = \langle 1,1,2,2\rangle)$$

$$(\{\bar{v}_{7,0} = \langle 1,0,1,2\rangle, \bar{v}_{7,1} = \langle 2,0,0,2\rangle, \bar{v}_{7,2} = \langle 2,1,0,1\rangle, \bar{v}_{7,3} = \langle 1,1,1,1\rangle\}, \bar{w}_7 = \langle 2,1,1,2\rangle)$$

$$(\{\bar{v}_{8,0} = \langle 0,0,2,2\rangle, \bar{v}_{8,1} = \langle 1,0,1,2\rangle, \bar{v}_{8,2} = \langle 1,1,1,1\rangle, \bar{v}_{8,3} = \langle 0,1,2,1\rangle\}, \bar{w}_8 = \langle 1,1,2,2\rangle)$$

Further, we describe how the situation changes as $k = 4$ and n grows, as we will need this construction to be explicit for later. For $n = 3, k = 4$, S_n is circumscibed by the rectangle formed by the set of points $\{\langle 0,0,3,3\rangle, \langle 3,0,0,3\rangle, \langle 3,3,0,0\rangle, \langle 0,3,3,0\rangle\}$. Along each line of this rectangle, there are two points in H_n (*and* S_n). We connect all of these points in a 4-by-4 grid of points, remniscent of Fig. 2.4, which also introduces the points in the interior of the outer rectangle. In particular, constructing a path along any fixed direction in this grid (that is, parallel to an axis) changes the points in a predictable manner. For example, consider a path from the upper-left point in Fig. 2.4 to the middle-left point to the bottom-left point. With each step, we subtract one from v_1 and add one to v_3, transforming $\langle 0,2,2,0\rangle$ into $\langle 0,0,2,2\rangle$ after two steps and encountering $\langle 0,1,2,1\rangle$ after one step. In the $n = 3, k = 4$ case, we can similarly move from $\langle 0,3,3,0\rangle$ to $\langle 0,0,3,3\rangle$ in three steps, sequentially encountering the points $\langle 0,2,3,1\rangle$ and $\langle 0,1,3,2\rangle$ along the path. This relationship holds as n increases to infinity, with the set of points $\{\langle 0,0,n,n\rangle$, $\langle n,0,0,n\rangle$, $\langle n,n,0,0\rangle$, $\langle 0,n,n,0\rangle\}$ circumscribing S_n.

Now we need to introduce a final concept in order to relate the solution spaces of *PlanarExactly-n* and *Exactly-*$\frac{k}{2}n$ (i.e., S_n(*PlanarExactly-n*) and S_n(*Exactly-*$\frac{k}{2}n$)). That is, as will be made explicit in the subsequent proof of the main theorem, it is easy to relate S_n(*PlanarExactly-n*) and S_n(*LinearExactly-n*), but consider the situation when, for example, $k = 4$ and we have a forbidden 4-rectangle $\{\bar{v}_0, \bar{v}_1, \bar{v}_2, \bar{v}_3\}$ for a point $\bar{w}$, where $\bar{v}_i \in S_n$(*PlanarExactly-n*), $\forall i$. By the proof of Theorem 13, we know that $\bar{w} \notin S_n$(*PlanarExactly-n*), but is it immediate that if the same $\bar{v}_i \in S_n$(*Exactly-*$\frac{k}{2}n$), $\forall i$, that they are a forbidden 4-rectangle for a point $\bar{w} \notin S_n$(*Exactly-*$\frac{k}{2}n$)? In particular, on the face of it, it could be the case that the coordinates in which the $\bar{v}_i$ differ from the $\bar{w}$ could differ in an inverse manner so that they *offset* and $\sum_i w_i = \frac{k}{2}n$ still.

For example, consider the following toy example. Let $k = 4$ and $n = 4$, and let the set $\{v_0 = \langle 3, 1, 1, 3\rangle, v_1 = \langle 1, 1, 3, 3\rangle, v_2 = \langle 1, 3, 3, 1\rangle, v_3 = \langle 3, 3, 1, 1\rangle\}$ form a forbidden 4-rectangle for the point $\bar{w} = \langle 1, 3, 1, 3\rangle$. In this case, note that for each $\bar{v}_i$, $v_0 + v_2 = 4 = n$ and $v_1 + v_3 = 4 = n$, so $\bar{v}_i \in S_n$(*PlanarExactly-n*), $\forall i$. Further, $w_0 + w_2 = 2 \neq n$, so $\bar{w} \notin S_n$(*PlanarExactly-n*). However, $\sum_i w_i = 8 = \frac{4}{2}4 = \frac{k}{2}n$, so $\bar{w} \in S_n$(*Exactly-*$\frac{k}{2}n$)!

However, we introduce the following technical lemma in order to show that when we encounter this problem in the case $k = 4$ if we "twist" the forbidden 4-rectangle (that is, permute the ordering of the parties in the protocol execution), we can *always* find a new point $\bar{w}' \notin S_n$(*Exactly-*$\frac{k}{2}n$) such that *the same* v_i are a forbidden 4-rectangle for this point as well, which will allow us to extend the lower bound for *PlanarExactly-n* to *Exactly-*$\frac{k}{2}n$ in the 4 party case immediately thereafter.

Twisting Lemma. *Let $k = 4$, and let $n \in \mathbb{N}$.*

Suppose there exists a point $\bar{w} = \langle w_0, w_1, w_2, w_3 \rangle$ and a set of distinct points $\mathcal{V} = \{\bar{v}_0 = \langle v_{0,0}, v_{0,1}, v_{0,2}, v_{0,3} \rangle, \bar{v}_1 = \langle v_{1,0}, v_{1,1}, v_{1,2}, v_{1,3} \rangle, \bar{v}_2 = \langle v_{2,0}, v_{2,1}, v_{2,2}, v_{2,3} \rangle, \bar{v}_3 = \langle v_{3,0}, v_{3,1}, v_{3,2}, v_{3,3} \rangle\}$ where $\bar{v}_i \in H_n, \forall i$, such that the following holds:

1. *$\mathcal{V}$ is a forbidden 4-rectangle for $\bar{w}$. That is, $v_{i,j} \neq w_j$ if $j = i \mod k$ or if $j = (i+1) \mod k$, for $0 \leq i < 4$; otherwise, $v_{i,j} = w_j$.*
2. *$\bar{v}_i \in S_n$(PlanarExactly-n), $\forall i$. That is, $\bigwedge_{j=0}^{\frac{k}{2}-1}(v_{i,j} + v_{i,(j+\frac{k}{2})} = n)$, $\forall i$.*
3. *$\bar{w} \notin S_n$(PlanarExactly-n). That is, $\exists j$ such that $w_{j \mod k} + w_{(j+\frac{k}{2}) \mod k} \neq n$.*
4. *$\bar{w} \in S_n$(Exactly-$\frac{k}{2}n$). That is, $\sum_j w_j = \frac{k}{2}n$.*

Then there exists a point $\bar{w}' = \langle w'_0, w'_1, w'_2, w'_3 \rangle$ and a permutation $\pi : \mathcal{V} \to \mathcal{V}$ such that the following holds:

A. *$\pi(\mathcal{V})$ is a forbidden 4-rectangle for $\bar{w}'$. That is, (we will abuse notation and say) $\pi(v_{i,j}) \neq w'_j$ if $j = i \mod k$ or if $j = (i+1) \mod k$, for $0 \leq i < 4$; otherwise, $\pi(v_{i,j}) = w'_j$.*
B. *$\pi(\bar{v}_i) \in S_n$(PlanarExactly-n), $\forall i$. That is, $\bigwedge_{j=0}^{\frac{k}{2}-1}(\pi(v_{i,j}) + \pi(v_{i,(j+\frac{k}{2})}) = n)$, $\forall i$.*
C. *$\bar{w}' \notin S_n$(PlanarExactly-n). That is, $\exists j$ such that $w'_{j \mod k} + w'_{(j+\frac{k}{2}) \mod k} \neq n$.*
D. *$\bar{w}' \notin S_n$(Exactly-$\frac{k}{2}n$). That is, $\sum_j w'_j \neq \frac{k}{2}n$.*

Proof. For $0 \leq j < 4$, let π be defined as follows:

$$\begin{aligned} \pi(v_{0,j}) &:= v_{1,j}, \\ \pi(v_{1,j}) &:= v_{0,j}, \\ \pi(v_{2,j}) &:= v_{3,j}, \\ \pi(v_{3,j}) &:= v_{2,j}. \end{aligned} \tag{2.3}$$

Recall that prior to applying π, we have the following situation:

$$\begin{aligned} \bar{w} &= \langle w_0, w_1, w_2, w_3 \rangle, \\ \bar{v}_0 &= \langle v_{0,0}, v_{0,1}, v_{0,2}, v_{0,3} \rangle, \\ \bar{v}_1 &= \langle v_{1,0}, v_{1,1}, v_{1,2}, v_{1,3} \rangle, \\ \bar{v}_2 &= \langle v_{2,0}, v_{2,1}, v_{2,2}, v_{2,3} \rangle, \\ \bar{v}_3 &= \langle v_{3,0}, v_{3,1}, v_{3,2}, v_{3,3} \rangle. \end{aligned} \tag{2.4}$$

By Condition (1) of the lemma, we can substitute as follows:

$$\begin{aligned} \bar{w} &= \langle w_0, w_1, w_2, w_3 \rangle, \\ \bar{v}_0 &= \langle v_{0,0}, v_{0,1}, w_2, w_3 \rangle, \\ \bar{v}_1 &= \langle w_0, v_{1,1}, v_{1,2}, w_3 \rangle, \\ \bar{v}_2 &= \langle w_0, w_1, v_{2,2}, v_{2,3} \rangle, \\ \bar{v}_3 &= \langle v_{3,0}, w_1, w_2, v_{3,3} \rangle. \end{aligned} \tag{2.5}$$

By Condition (2) of the lemma, we can substitute as follows:

$$\begin{aligned}
\bar{w} &= \langle w_0, w_1, w_2, w_3 \rangle, \\
\bar{v}_0 &= \langle (n - w_2), (n - w_3), w_2, w_3 \rangle, \\
\bar{v}_1 &= \langle w_0, (n - w_3), (n - w_0), w_3 \rangle, \\
\bar{v}_2 &= \langle w_0, w_1, (n - w_0), (n - w_1) \rangle, \\
\bar{v}_3 &= \langle (n - w_2), w_1, w_2, (n - w_1) \rangle.
\end{aligned} \tag{2.6}$$

Now we apply the permutation π and set $\bar{w}' := \langle (n - w_2), w_1, (n - w_0), w_3 \rangle$, which gives:

$$\begin{aligned}
\bar{w}' &:= \langle (n - w_2), w_1, (n - w_0), w_3 \rangle, \\
\pi(\bar{v}_0) = \bar{v}_1 &= \langle w_0, (n - w_3), (n - w_0), w_3 \rangle, \\
\pi(\bar{v}_1) = \bar{v}_0 &= \langle (n - w_2), (n - w_3), w_2, w_3 \rangle, \\
\pi(\bar{v}_2) = \bar{v}_3 &= \langle (n - w_2), w_1, w_2, (n - w_1) \rangle, \\
\pi(\bar{v}_3) = \bar{v}_2 &= \langle w_0, w_1, (n - w_0), (n - w_1) \rangle.
\end{aligned} \tag{2.7}$$

Now we can prove Implication (A), that $\pi(\mathcal{V})$ is a forbidden 4-rectangle for $\bar{w}'$, or in other words, $\pi(v_{i,j}) \neq w'_j$ if $j = i \mod k$ or if $j = (i+1) \mod k$, for $0 \leq i < 4$, and otherwise, $\pi(v_{i,j}) = w'_j$.

In particular, we have the equalities of (A) immediately from Equation (2.6), and what remains to be shown are the inequalities. From Condition (3), we know that $\exists j$ such that $w_{j \mod k} + w_{(j+\frac{k}{2}) \mod k} \neq n$. However, due to Condition (4), which says $\sum_j w_j = \frac{k}{2} n = 2n$ (where the second equality is due to $k = 4$), we can extend Condition (3) to say that $\forall j$, $w_{j \mod k} + w_{(j+\frac{k}{2}) \mod k} \neq n$. We prove the extension by showing if $\exists$ a *unique* j such that $w_{j \mod k} + w_{(j+\frac{k}{2}) \mod k} \neq n$ then we have a contradiction. Since $k = 4$, there are two cases:

- **Case 1:** Suppose $w_0 + w_2 = n$ and $w_1 + w_3 \neq n$. Then by Condition (4), we have

$$w_0 + w_1 + w_2 + w_3 = 2n$$

Substituting $w_0 + w_2 = n$ gives,

$$n + w_1 + w_3 = 2n$$

$$w_1 + w_3 = n$$

which contradicts the supposition, so Case 1 cannot hold.

- **Case 2:** Suppose $w_0 + w_2 \neq n$ and $w_1 + w_3 = n$. Then by Condition (4), we have

$$w_0 + w_1 + w_2 + w_3 = 2n$$

Substituting $w_1 + w_3 = n$ gives,

$$n + w_0 + w_2 = 2n$$

$$w_0 + w_2 = n$$

which contradicts the supposition, so Case 2 cannot hold.

Therefore, substituting $k = 4$, $\forall j$, $w_{j \mod 4} + w_{(j+2) \mod 4} \neq n$. Returning to Equation (2.6), we find that every inequality that remains to be shown for (A) to hold are all of the form $w_{j \mod k} \stackrel{?}{\neq} (n - w_{(j+2) \mod k})$. Equivalently, including $k = 4$ we rewrite to get $w_{j \mod 4} + w_{(j+2) \mod 4} \stackrel{?}{\neq} n$, which is true $\forall j$. So we have (A).

We also immediately get Implication (B) from Equation (2.6). That is, we want to know if $\bigwedge_{j=0}^{\frac{k}{2}-1}(\pi(v_{i,j}) + \pi(v_{i,(j+\frac{k}{2})}) = n)$, $\forall i$ holds. However, all of the $(\pi(v_{i,j}) + \pi(v_{i,(j+\frac{k}{2})})$ for $j = 0$ and $j = 1$ are of the form $x + (n - x)$ (resp. $(n - x) + x$) for some coordinate value x. In all cases, the x and $-x$ cancel, leaving n. So we have (B).

Recall that by the definition of $\bar{w}'$ we have the following values for the coordinates of $\bar{w}'$:

$$\begin{aligned} w'_0 &= (n - w_2) \\ w'_1 &= w_1 \\ w'_2 &= (n - w_0) \\ w'_3 &= w_3 \end{aligned} \tag{2.8}$$

From the proof of (A), we know that $w_1 + w_3 \neq n$, and consequently from Equation (2.7), we have that $w'_1 + w'_3 \neq n$. Thus, setting $j = 3$ in the statement of Implication (C) shows that it holds.

Finally, we will prove the key fact, Implication (D). Suppose $\sum_i w_i = \sum_i w'_i$. Then we have the following:

$$\begin{aligned} w_0 + w_1 + w_2 + w_3 &= (n - w_2) + w_1 + (n - w_0) + w_3 \\ w_0 + w_1 + w_2 + w_3 &= 2n + w_1 + w_3 - w_0 - w_2 \\ w_0 + w_2 &= 2n - w_0 - w_2 \\ 2w_0 + 2w_2 &= 2n \\ w_0 + w_2 &= n \end{aligned} \tag{2.9}$$

However, this is a contradiction, since we know $w_0 + w_2 \neq n$ from the proof of (A). Therefore, $\sum_i w_i \neq \sum_i w'_i$. But $\sum_i w_i = \frac{k}{2}n$ by Condition (4), so $\sum_i w'_i \neq \frac{k}{2}n$, and we have (D). ■

We demonstrate how the Twisting Lemma operates by applying it to the previous example. Recall that $k = 4, n = 4$, that the set $\{\bar{v}_0 = \langle 3, 1, 1, 3\rangle, \bar{v}_1 = \langle 1, 1, 3, 3\rangle, \bar{v}_2 = \langle 1, 3, 3, 1\rangle, \bar{v}_3 = \langle 3, 3, 1, 1\rangle\}$ forms a forbidden 4-rectangle for the point $\bar{w} = \langle 1, 3, 1, 3\rangle$, that $\bar{v}_i \in S_n(\textit{PlanarExactly-}n)$, $\forall i$, that $\bar{w} \notin S_n(\textit{PlanarExactly-}n)$, and that $\bar{w} \in$

$S_n(\textit{Exactly-}\frac{k}{2}n)$. As a result, the scenario fits the conditions for the Twisting Lemma, so we "twist" the forbidden 4-rectangle, giving the set $\{\pi(\bar{v}_0) = \langle 1,1,3,3\rangle, \pi(\bar{v}_1) = \langle 3,1,1,3\rangle, \pi(\bar{v}_2) = \langle 3,3,1,1\rangle, \pi(\bar{v}_3) = \langle 1,3,3,1\rangle\}$, and we let the point $\bar{w}' = \langle (n - w_2), w_1, (n - w_0), w_3\rangle = \langle (4-1), 3, (4-1), 3\rangle = \langle 3,3,3,3\rangle$. Observe that the set $\{\pi(\bar{v}_i)\}, \forall i$, forms a forbidden 4-rectangle for $\bar{w}'$. Further, for each $\pi(\bar{v}_i)$, $\pi(v_0) + \pi(v_2) = 4 = n$ and $\pi(v_1) + \pi(v_3) = 4 = n$, so $\pi(\bar{v}_i) \in S_n(\textit{PlanarExactly-}n)$, $\forall i$. Additionally, $w'_0 + w'_2 = 6 \neq n$, so $\bar{w}' \notin S_n(\textit{PlanarExactly-}n)$. And finally, $\sum_i w'_i = 12 \neq \frac{4}{2}4 = \frac{k}{2}n$, so $\bar{w}' \notin S_n(\textit{Exactly-}\frac{k}{2}n)$.

Now we can prove our main theorem:

Theorem (Main). *Instances of Monochromatic-Rectangle-Free Grid Coloring (i.e.,* RECT-EXTEND*) that are "tight"* NO *instances for any fixed c, or c-minimal* (see Section 1.3.2 for a precise definition), *provide lower bounds for the multiparty communication complexity of the predicates Exactly-*$\frac{k}{2}n$*, PlanarExactly-n, and LinearExactly-n for the case* $k = 4$.

Proof. By Theorem 13, we know that the communication complexity of any k-party protocol for *PlanarExactly-n* is bounded below by the logarithm of the minimum number of colors required to color the points of S_n (that is, $S_n(\textit{PlanarExactly-}n)$) so that no forbidden k-rectangles are colored monochromatically, i.e. $\mu_k(n)$. By Lemmas 13.1 and 13.2, we know that $S_n(\textit{PlanarExactly-}n) \subsetneq S_n(\textit{Exactly-}\frac{k}{2}n)$ and $S_n(\textit{PlanarExactly-}n) \subsetneq S_n(\textit{LinearExactly-}n)$. Therefore, all of the forbidden k-rectangles in $S_n(\textit{PlanarExactly-}n)$ are also in $S_n(\textit{Exactly-}\frac{k}{2}n)$ and in $S_n(\textit{LinearExactly-}n)$.

First we will show that (1) $\mu_k(n)$ provides a similar, general lower bound for the communication complexity of *LinearExactly-n*, then we will show that (2) $\mu_4(n)$ provides a lower bound for the communication complexity of *Exactly-*$\frac{k}{2}n$ in the 4 party case, and then (3) we will relate Monochromatic-Rectangle-Free Grid Coloring

and $\mu_4(n)$.

1. Suppose the communication complexity of any k-party protocol for *LinearExactly-n* was less than the logarithm of the minimum number of colors required to color the points of S_n(*PlanarExactly-n*) so that no forbidden k-rectangles are colored monochromatically. Then there would exist a k-party protocol that computes *LinearExactly-n* with cost less than $\log(\mu_k(n))$. However, this would imply that a forbidden k-rectangle of S_n(*LinearExactly-n*) is colored monochromatically – namely, one of the rectangles in S_n(*PlanarExactly-n*) that is also in S_n(*LinearExactly-n*). This, in turn, implies there exists a point $\bar{w}$ that is also colored identically. However, by definition of a forbidden k-rectangle, this $\bar{w}$ must differ from each of the $\bar{v}_i$ of the rectangle in precisely one coordinate of the pair $(v_i, v_{i+\frac{k}{2}})$. But since $\bar{v}_i \in S_n(\textit{PlanarExactly-n}), \forall i$, it is the case that $\bigwedge_{i=0}^{\frac{k}{2}-1}(v_i + v_{i+\frac{k}{2}} = n)$. If you change one and only one coordinate's value in all of the respective summations, then none of them will sum to exactly n. Therefore, it cannot be the case that $\bigvee_{i=0}^{\frac{k}{2}-1}(v_i + v_{i+\frac{k}{2}} = n)$, so $\bar{w} \notin S_n(\textit{LinearExactly-n})$, which is a contradiction – that is, no such k-party protocol exists. Therefore, the multiparty communication complexity of *LinearExactly-n* is $\Omega(\log(\mu_k(n)))$.

2. Similarly for *Exactly-*$\frac{k}{2}n$ when $k = 4$, as we know that all of the forbidden 4-rectangles in S_n(*PlanarExactly-n*) are in S_n(*Exactly-*$\frac{k}{2}n$), it remains to show that there exists a $\bar{w}$ for each of these rectangles that is not in S_n(*Exactly-*$\frac{k}{2}n$) as the existence of one such $\bar{w}$ per rectangle would cause protocols costing less than $\log(\mu_4(n))$ to be invalid for *Exactly-*$\frac{k}{2}n$ when $k = 4$. Fix any such rectangle and consider its associated $\bar{w}$. If $\bar{w} \notin S_n(\textit{Exactly-}\frac{k}{2}n)$, we are done. If, however, $\bar{w} \in S_n(\textit{Exactly-}\frac{k}{2}n)$, then we "twist" the rectangle via the Twisting Lemma, and we have a $\bar{w}'$ for which the rectangle is still a forbidden 4-rectangle and such that $\bar{w}' \notin S_n(\textit{Exactly-}\frac{k}{2}n)$.

3. Finally, for $k = 4, n \geq 1$, we simply substitute every *point* of each respective predicate's S_n with the geometrically corresponding *cell* from an instance of Monochromatic-Rectangle-Free Grid Coloring. For c-minimal $G_{n,n}$ (or the smallest $G_{n,n}$ not contained in any grid of OBS_c), $c + 1 = \mu_4(n)$ by definition. In particular, the value of $\mu_4(n)$ increases by 1 *precisely* each time n crosses the "boundary" of a c-minimal grid size, for any c. ■

2.3.3 The Ramsey-Communication Conjecture

Based on the above study, we now conjecture that the relationship between Ramsey-theoretic mathematics and lower bounds in communication complexity extends much further. First, we generalize the (as-yet informal) notion of Shape-Free Grid Coloring.

In particular, define *Shape-Free Space Coloring* as the class of computational problems such that every problem in the class has the following two properties:

1. Let $d \geq 3$. Then the problem involves coloring points of a finite, discrete, d-dimensional space such that no *shape* (an arbitrary but fixed, geometrically-defined relationship between points in the space) is colored monochromatically.

2. Van der Waerden's Theorem applies to the problem by a projection from the natural numbers to points in the colorable space.

We conjecture the following:

The Ramsey-Communication Conjecture. *Fix any problem in Shape-Free Space Coloring. Then c-minimal instances of this problem provide lower bounds on the multiparty communication complexity of some predicate.*

2.4 Questions for Further Research

- We did not give a proof that Monochromatic-Rectangle-Free Grid Coloring provides a *tight* lower bound for any predicate. In the case of the predicates present in this thesis, especially for *PlanarExactly-n* and *LinearExactly-n*, this seems unlikely. For *PlanarExactly-n* and *LinearExactly-n*, it seems reasonable that they would reduce to a multiparty analog of the Equality predicate, which has a tight two-party communication complexity of $\Theta(n)$. In the case of *Exactly-*$\frac{k}{2}n$ (or perhaps more generally, *Exactly-m* for $m \in [0, kn]$), it seems less obvious that the "correct" lower bound is $\Omega(\frac{k}{2}n)$ (i.e. the amount of bits required to give any one party all of the available information), even though we could not provide a matching upper bound of $O(\log(\mu_k(n)) + c)$ for constant c. What is the exact communication complexity of *Exactly-*$\frac{k}{2}n$? Or similarly, does there exist a predicate where Monochromatic-Rectangle-Free Grid Coloring provides a matching upper and lower bound (i.e. for which Monochromatic-Rectangle-Free Grid Coloring provides an *answer*)?

- Can we provide a stronger, more explicit characterization of the relationship between Ramsey-theoretic mathematics and lower bounds in communication complexity? In other words, is the Ramsey-Communication Conjecture true? Can it be refined to be more precise? (Does it in fact need to be?)

- Can we unconditionally prove Theorem 12? In particular, is the NAE for Robust for Grids Assumption true?

- Where does the computational complexity of computing c-minimal grids in various Shape-Free Grid Coloring problems ultimately lie? Given the NAE is Robust for Grids Assumption, we know there are two possibilities: (1) the problem is NP-complete, or (2) the problem is NP-intermediate.

– *What would an* NP*-completeness proof for Shape-Free Grid Coloring look like?* The straightforward answer is a standard reduction from an NP-complete problem to some (or all) variants of Shape-Free Grid Coloring. One possibility would be a reduction from E4-Set-Splitting or a similar problem to RECT-EXTEND. In Ek-Set-Splitting, known to be NP-complete for $k \geq 3$ via [14], seems to capture the relational structure of Shape-Free Grid Coloring, sans the structural restriction to a grid-shaped hypergraph. This is in much the same vein to how CSP($\mathcal{G}$) captures the structural restrictions but requires arbitrary relations. In essence, we have two apparently hard problems whose intersection is Shape-Free Grid Coloring, which is intersting but not sufficient for a proof of Shape-Free Grid Coloring's hardness. How can this line of thinking be extended?

– *What would an* NP*-intermediateness proof for Shape-Free Grid Coloring look like?* In general, one should not hope to easily prove the unconditional existence of NP-intermediate problems (not without separating P and NP first!), but given an assumption of the ETH, perhaps some progress is possible. For instance, under the ETH, giving an explicit subexponential time algorithm for RECT-EXTEND would be a first step (though not a sufficient one for technical reasons: the ETH's time complexity claim only directly regards SAT). Another avenue would be to show that "*something bad*" happens (like a collapse of the polynomial hierarchy) if some predicate's lower bound, $\mu_k(n)$ requires the same degree of complexity to compute for all values of k. This approach would, in essence, be initial evidence of an infinite hierarchy of increasingly more complex problems as k grows, matching the known consequences of Ladner's Theorem [30] for NP-intermediate problems.

- And further, what is the complexity of obtaining various approximation guaran-

tees for problems in Shape-Free Grid Coloring? As mentioned before, computing relatively close approximations to the 17x17 4-Color Problem appears, in practice, to be much easier than exactly computing a full monochromatic-rectangle-free solution[39]. On the other hand, the similarity between Shape-Free Grid Coloring and E4-Set-Splitting raises the interesting possibility of a PCP-based inapproximability threshold in the vein of [22]. Without an unconditional, fixed complexity for even RECT-EXTEND, this question seems wide open.

Chapter 3

The 17x17 4-Color Problem

3.1 Introduction

In this chapter, we describe an approach, based on empirical evidence, that is designed to solve the 17x17 4-Color Problem. The total runtime of the approach is expected to take under *one year* to complete and is guaranteed to either find a monochromatic-rectangle-free 4-coloring of the 17x17 grid or show that no such coloring exists. To our knowledge, this is – by far – the fastest documented approach that guarantees a solution. At the same time, it falls just outside of the realm of practicality in terms of actually being implemented. To actually attack the problem, some improvements need to be made to lower the runtime requirements.

Recall that the 17x17 4-Color Problem is defined as follows:

The 17x17 4-Color Problem. A *rectangle of* $G_{17,17}$ is a set of the form $\{(i,j),(i+a,j),(i,j+b),(i+a,j+b)\}$ for constants i, j, a, b such that the rectangle is contained in $G_{17,17}$. Given four colors, is there a way to color every element of $G_{17,17}$ such that $G_{17,17}$ does not contain a rectangle of all the same color?

It is particularly worthwhile to note that many *approximate* colorings, where the goal to is minimize the number of monochromatic rectangle constraints that are violated by a color assignment, are known and (relatively speaking) easy to find in practice[39]. The best known approach for approximate colorings is *simulated annealing* [27], which draws inspiration from the metallurgical practice of annealing, or applying heat to metals to refine them.

Simulated annealing is a randomized heuristic for finding approximations to a

global optimization function over a large, discrete search space. In general, simulated annealing begins at an arbitrary (or randomized) initial state, and at each iteration, considers some new state that is "close" to the current state, where the distance metric is chosen on a per-application basis. The algorithm then probabilistically chooses whether to accept or reject a transition from the current state to the candidate neighbor-state as a function of the difference in the approximation value of the two states and a global parameter T, called the temperature. Over time, T decreases from ∞ to 0, at which point the algorithm terminates.

In the case of simulated annealing for monochromatic-rectangle-free colorings (and the 17x17 4-Color Problem in particular), here is a candidate implementation in concept. First, we begin with a random coloring of the 17x17 grid. We set some parameter k to count the number of total iterations and terminate at some point in the future depending on how much time we have. On each iteration, we do the following. First, we check the current grid for violated rectangle constraints. We then chose one of those at uniform random. For the given violated rectangle constraints, we then choose one of its four constituent vertices at uniform random. We then select a new, candidate color for that vertex at uniform random. Then we compare the *percent* of violated constraints of the total number of constraints between the current grid coloring and the current grid coloring with the new, candidate color for our chosen vertex. We then probabilistically select a threshold at which we will accept the new coloring, depending on the previously computed percent difference between the current and new colorings and the value of k. For example, we might randomly choose "If the new coloring violates 1% fewer constraints than our current coloring or less, we will accept." Similarly, we could choose to accept a *worse* approximate coloring on occasion, in the hope that it will help break out of a local minimum. Finally, if we switch, we record the best coloring so far, increment k, and repeat the process.

However, we believe that it is worthwhile to primarily investigate approaches that allow one to show that no monochromatic-rectangle-free coloring of $G_{17,17}$ exists. Our motivation for this belief comes from two facts: (1) an approximate coloring of $G_{17,17}$ brings us no closer to resolving the proper elements of OBS_4, and (2) very good approximate colorings are already known. For an example of the latter, here is an almost-4-coloring of $G_{17,17}$, due to Rohan Puttgunta[35], that contains one uncolorable cell (marked by "+") but no monochromatic rectangles.

00000111122223333
01111122223333000
02222133320003111
03333100021113222
10123212332300301
11032221033210032
12301230130120123
13210203231030210
20231313002011312
21320320303101021
22013331200231130
23102302101321203
30312010212132320
31203023113022013
32130032010312102
33021001311202231
0123+012301230123

Therefore, we hypothesize the following and aim to either prove or disprove it:

Hypothesis. There is no monochromatic-rectangle-free 4-coloring of $G_{17,17}$.

In the following sections, we first discuss some of our new results in monochromatic-*square*-free c-coloring as the inspiration for our approach to the 17x17 4-Color Problem, then we discuss the plan itself from a high-level view before digging deeper into the technical aspects of its two phases. Finally, we discuss areas for improvement and some ideas for getting there.

3.2 An aside: Monochromatic-Square-Free c-coloring

Our results in monochromatic-square-free c-coloring include a 2-coloring of $G_{13,13}$ and a 3-coloring of $G_{39,39}$. A *monochromatic-square-free c-coloring of* $G_{n,m}$ is an assignment of colors in $\{0, ..., c-1\}$ to the vertices of $G_{n,m}$ such that no vertices arranged in a square are colored identically. In this context, a *square* is a set of four vertices $\{(i,j),(i+a,j),(i,j+a),(i+a,j+a)\}$ for constants i, j, and a such that each pair, (x,y), is a vertex of $G_{n,m}$. We first present the colorings and then discuss the methods and consequences.

Here is a monochromatic-square-free 2-coloring of $G_{13,13}$:

0000001001111
0101100101010
0011001111001
1110100010011
1011111001000
0110010011110
1101001010100
1000011110010
1011000100111
0001110010101
0101011000011
1100010101001
0110111100100

And here is a monochromatic-square-free 3-coloring of $G_{39,39}$:

000000000000000000000000000000000000000
010101010101010101010101010101010101010
001100110011001100110011001100110011001
011201120112011201120112011201120112011
000011210000112100001121000011210000112
010112220101122201011222010112220101122
002121120021211200212112002121120021211
021111221000000011121220021111221000000
000000001111222100000000111221210020101
010102201222110010201011101020201011102
002100112112122200210012112212210001220
022101012211221222221000000100011221121
000012111212211200101202120221010010022
010211121221221112100002110020011211001
011010201020102001011121211222101012020
001111021111122220001220120001101120012
010001010100010010222122102110112101002
000200100010202111121112121202120000200
011002222111002022201121100011212201120
100220211121220112012200102200101202120
001022100212210012021211220210010022212
021211212222101021002212020202011221101
100112011100202121022102220010002112100
101022010020000210200100121122222121110
000200011212101112122120222011100010211
020001102001110122211200112112001022002
002122120221001021120001211202021012122
101011212002121220002011202020121220112
111100120020122020110220010102101102211
222120211002001200010002011012022122121
020010222011111221022112101011102202220
100202000212021002101010112012002111212
001110020122011011110112202200102100022
021220100100202101202221010201012110112
110202021022010202211211100101001010202
211221100120111200001122210120020220200
012000201110121111011020022202222202210
002021112021120102110002211220101210221
100002102000222010200122112100021111020

The first coloring was generated by a simple backtracking algorithm implemented in C++ compiled using the GNU C++ compiler, version 4.5.2[17]. It was executed purely sequentially on a single core running at 3.00 GHz and took approximately 174 hours of consecutive execution to find. Here is a pseudocode implementation:

```
procedure search ( grid : an n × m array initialized to all -1 ,
                   c : a number of allowed colors )
    x ← 0, y ← 0
    while true
        grid_{x,y} ← grid_{x,y} + 1
        if check(grid, x, y)
            if not goToNext(x, y)
                output grid
            end if
        else if grid_{x,y} = c - 1
            if not backtrack(grid, x, y)
                output false
            end if
        end if
    end while
end procedure
```

The further, listed procedures – **check**, **goToNext**, and **backtrack** – may be implemented as desired. However, some optimizations are possible. In particular, **check** needs to only check for violate square constraints for those vertices that have already been colored up to that point in the search. **goToNext** can specify any ordering of the vertices, but in our case, a simple row-major ordering was used. The

procedure should modify its inputs (x, y) to be (x', y'), representing the next vertex in the ordering (if possible), and return false if there are no new vertices to "move" to or true otherwise. **backtrack** is simply the inverse operation of **goToNext**, reversing whatever ordering of vertices is specified in **goToNext**, modifying its input to represent the immediately previous vertex in the ordering if possible and returning false if given as input the coordinates of the initial vertex or true otherwise. Additionally, **backtrack** will reset the value of the vertex it is passed (that is, its color) to -1.

The second coloring (of $G_{39,39}$) was generated in a slightly modified manner. Instead of running the backtracking search from an empty grid (i.e. one without any colors assigned), the initial grid coloring was seeded with a legal coloring of a lower-order grid. In particular, the overall process was done via *bootstrapping* – solutions to one size of grid were given as the starting point for a backtracking search using the previously mentioned algorithm for a larger grid later on. Initially, of course, colorings were generated from an uncolored grid: first for $G_{11,11}$, then again for $G_{12,12}$, and again for $G_{13,13}$, and so forth. In the mid-20s – around $G_{25,25}$ – the runtime of solving each instance from scratch began to noticably slow down. We then began inserting colorings of smaller grids into the lower-right quadrant of the larger grids (flush against the axes) and running the backtracking search from that point onward. Note that this implies the algorithm would immediately backtrack into the seeded coloring and change it. For example, when searching for a square-free coloring for $G_{30,30}$, the lower-right quadrant of the grid was seeded with the square-free 3-coloring of $G_{15,15}$, and so forth (we note that the choice of seeded grid coloring was done in an *ad hoc* manner; in other words, completely arbitrary choices were made with each iteration). The entire process of generating the square-free 3-coloring of $G_{39,39}$ was completed in approximately 24 hours via this method. The final coloring obtained – that of $G_{39,39}$ – took approximately three hours to generate from a seeding of the 3-coloring for $G_{38,38}$.

Admittedly, there is some luck associated with arriving at colorings in the manner we did, due to the non-rigorous, intuitive way in which input seedings were chosen. At the same time, one important observation is that a row-major ordered backtracking search finds colorings in *lexicographical* order. That is, for some fixed number of colors c, we can interpret every coloring as a base-c number in order to gain some idea of how far through the candidate solution space of each instance that the search had to proceed. For example, interpreting the coloring of $G_{39,39}$ as a base-3 number gives a decimal value of approximately 10^{199}, which is an almost-infintesimal percentage of the total search space $3^{39,39} = 3^{1521}$, or approximately 10^{-525}. Even then, a huge percentage of the space actually searched was telescoped over by the backtracking algorithm, since every time a violated square constraint was found – especially early in the lexicographical ordering – some large sequence of possible colorings was skipped. Even more were omitted by initializing the search from the middle of the ordering by providing seeded inputs.

These two results are surprising for different reasons. In the case of the 2-coloring for $G_{13,13}$, it was believed that $G_{10,10}$ was the limit for square-free 2-colorings[41]. Previous work that arrived at the 2-coloring of $G_{10,10}$, primarily by Jim Purtilo at Maryland-College Park, focused on a different approach. Beginning with $G_{2,2}$ and scaling upwards, an exhaustive, brute-force enumeration was performed of each legal coloring for the grids $G_{n,n}$. At each new grid size, the grid was seeded with every possible coloring of the previous size, based on the observation that any $n-1$-by-$n-1$ subset of $G_{n,n}$ must be a legal coloring of $G_{n-1,n-1}$. (We wish to attribute the inspiration for seeding the search with previous grid colorings leading up to the discovery of the 3-coloring of $G_{39,39}$ to him.)

In the case of the 3-coloring for $G_{39,39}$, given how apparently difficult the 17x17 4-Color Problem is, we find it striking how quickly a legal coloring was found in a massively larger candidate solution space. That is, one of the most-cited, intuitive

reasons for believing that the 17x17 4-Color Problem is difficult is that there are $4^{17\times17} = 4^{289} \simeq 10^{174}$ possible colorings one has to check (or, respectively, telescope over via backtracking). On the other hand, there are $3^{39\times39} = 3^{1521} \simeq 10^{725}$ possible 3-colorings of $G_{39,39}$, and $10^{725} \gg 10^{174}$. There are many possible reasons that the above approach so quickly found a 3-coloring square-free solution for $G_{39,39}$; we list a few here:

1. *Pure luck.* That is, there just *happened* to be a single solution very early in the lexicographical ordering of all square-free 3-colorings of $G_{39,39}$. We cannot rule out this possibility, but given that solutions were found similarly quickly for $G_{10,10}$ through $G_{39,39}$, we feel there may be a better explanation.

2. *There are many solutions, by percentage of possible colorings.* That is, it *could* be possible that $G_{13,13}$ and $G_{39,39}$ are nowhere near the limit of what is square-free colorable with 2 and 3 colors, respectively. We tested this hypothesis by running a pure Monte Carlo search for square-free colorings – i.e. randomly assigning colors to *all* vertices in the grid, checking for violations, and repeating until a legal coloring was found. However, over 2 days of execution failed to find a single, legal 3-coloring for even $G_{10,10}$. While this may not be enough to rule out the possibility in theory, there is a distinct difference in practical runtimes.

3. *Lexicographical search exploits an important property of shape-free colorings.* While this would be very exciting in terms of solving the 17x17 4-Color Problem, applying the same algorithm to search for monochromatic-rectangle-free colorings of relatively low-order grids did not give the expected results for this hypothesis. It took approximately one day to find a legal, monochromatic-rectangle-free 4-coloring of $G_{12,12}$.

4. *Lexicographical search exploits an important property of square-free colorings.* In some sense, this is the only hypothesis remaining that we find plausible.

While we did not extensively explore this ourselves, we believe it would be an interesting path for further research to determine and characterize the exact distribution of legal, square-free colorings with respect to lexicographical ordering of all possible colorings by enumerating all such solutions through backtracking, brute-force, or similar approaches.

Taken altogether, we believe this provides empirical value to the claim that the backtracking approach is often excellent for finding a solution, if one exists. Of course, if a solution does not exist for a problem, backtracking defaults to a slightly improved brute-force search, and may often require a runtime on the order of the candidate solution space itself to find an answer. In the remainder of the chapter, we present a computational plan based on a similar, long-term-oriented, backtracking approach to the 17x17 4-Color Problem using high-performance computing resources and carefully analyze the expected runtime and memory requirements.

3.3 Computational Plan

One of the greatest, if not *the* greatest, challenges in taking a theoretically-designed approach and implementing it on a supercomputer comes in the form of the processor-hours and memory allowances available on any given, actual machine. For instance, if an algorithm is intended to run in multiple phases, the first phase has a projected runtime of, say, 250,000 processor-hours, and the systems available are a 10,000-core machine and a 1,000-core machine, but access to the first machine is only given in allotments of 200,000 processor-hours at a time, there is a hard decision. Is it worthwhile to modify how the phases of execution are divided up so that the first phase can run an order of magnitude faster, or will breaking the overall problem up into smaller pieces cause too long of a delay?

The overall structure of the computational plan is to first use a backtracking-style, brute-force search to enumerate all legal, monochromatic-rectangle-free colorings of

some lower-order grid size. Then, to bootstrap towards the larger grid sizes, we treat each legal coloring as a valid color for the larger grid and repeat the procedure. Consider the following toy example:

> Suppose we wanted to find all monochromatic-rectangle-free k-colorings of $G_{8,8}$, for an arbitrary integer $k > 1$. Our approach will be to first enumerate every possible monochromatic-rectangle-free k-coloring of $G_{4,4}$ from scratch. Suppose that such a search is projected to take 50,000 processor-hours and generate 50 GBs of data – the file, or series of files, containing all legal k-colorings of $G_{4,4}$. In the first phase, we need a machine on which we are allowed at least 50,000 processor-hours and that has access to an output hard drive with 50 GB of space. Fortunately, the RAM requirements of the first phase are quite low, since the only thing we have to hold in RAM is a 4-by-4 array of integers. We run the search, generate 50 GB of data, and proceed to the next phase.
>
> In the next phase, we treat each 4-by-4 k-coloring as a *single* "color" (or, "colored tile") in the search for a monochromatic-rectangle-free k-coloring of $G_{8,8}$. Suppose, then, in our 50 GB hard drive, we have 10^{20} such colorings. In essence, we perform a new search for a monochromatic-rectangle-free 10^{20}-coloring of $G_{2,2}$, since covering $G_{8,8}$ with colorings of $G_{4,4}$ takes four such colorings, one for the top-left, top-right, bottom-left, and bottom-right quadrants respectively. Of course, the checking procedure will still treat the grid as $G_{8,8}$ with 4 colors available per vertex (rather than 10^{20} colorings available per quadrant). In this case, the runtime estimate in terms of processor-hours becomes a major constraint as well as the RAM requirements (since, to avoid the dramatic slow-down associated with multiple hard drive reads, we have to load all 10^{20} colorings into RAM simultaneously to perform backtracking to even compare the

possibilities for the top-left and top-right quadrants), while the output hard drive space should, in theory, become much less of a problem. That is, assuming $G_{8,8}$ is at or close to the threshold of legal k-colorings, there should be very few such colorings to output.

It is important to know, when reading the above example, that such a backtracking search is inherently embarassingly parallelizable. An *embarassingly parallelizable* algorithm is defined in folklore as one where, under Amdahl's Law, $Speedup \simeq n$ for number of processes, n. Amdahl's law [5] states:

$$Speedup = \frac{1}{r_s + \frac{r_p}{n}} \tag{3.10}$$

where $r_s + r_p = 1$ and r_s represents the ratio of the sequential portion in the algorithm.

Theorem. *For a backtracking-style algorithm solving the $n \times m$ c-Color Problem, as described above, $r_p \simeq 1$.*

Proof. For a given number of processes n, we parallelize the search in the following manner. Compute the size of total candidate solution space, S, by $|S| = c^{n \times m}$ for colors c and constants n and m as dimensions of $G_{n,m}$. Compute the set of n integers $I = \left\{0, |S|\lceil\frac{1}{n}\rceil, |S|\lceil\frac{2}{n}\rceil, ..., |S|\lceil\frac{n-1}{n}\rceil\right\}$. Convert each of the integers $i \in I$ to its base-c representation, i_c. Pass one i_c to each of the n processes respectively as initial, starting points, and pass each of the $(i+1)_c$ to each of the same, respective n processes as final stopping points. The n^{th} process halts at $|S|$. Since we can do this procedure for any value of n up to $|S|$, $r_p \simeq 1$. ■

Corollary. *For a backtracking-style algorithm solving the $n \times m$ c-Color Problem, as described above, Speedup $\simeq n$, which implies it is embarassingly parallelizable.*

Proof. We apply Amdahl's Law:

$$\begin{aligned} Speedup &= \frac{1}{r_s + \frac{r_p}{n}} \\ &\simeq \frac{1}{0 + \frac{1}{n}} \\ &\simeq n \end{aligned}$$ ■

As a result, a nice feature of this approach is that the number of cores on a given machine running an implementation of this approach is essentially independent of the problem's constraints. If run a 1,000-core machine, we can expect roughly a 10^3-times speedup, or if run on a 10,000-core machine, we can expectly roughly a 10^4-times speedup, and so forth. Therefore, once we have a good, empirically validated projection of the number of processor-hours required by this approach, the machine in question could be any machine, where more cores means lower runtime but not more hassle.

Overall, the computational plan involves two primary phases. The first phase is a backtracking-styled, brute-force enumeration of all legal 4-colorings of $G_{8,8}$. The second phase is a backtracking-styled, brute-force enumeration of all legal 4-colorings of $G_{16,16}$, where we use the legal colorings of $G_{8,8}$ to cover quadrants of $G_{16,16}$ at a time. Finally, we use a final backtracking step to search for extensions from legal colorings of $G_{16,16}$ to legal colorings of $G_{17,17}$. However, we assume the number of legal colorings of $G_{16,16}$ will be small – likely many orders of magnitude smaller than the number of legal colorings of $G_{8,8}$, since $G_{17,17}$ is quite close to members of, or even a proper member of, OBS_4. As such, we now describe the two primary phases of execution and assume that the final step will be a relatively trivial clean-up phase.

3.3.1 Phase 1

For the first phase, we will use backtracking to enumerate all legal, monochromatic-rectangle-free 4-colorings of $G_{8,8}$. To simulate the typical execution of this phase, we implemented a serial version of the program and ran the code for 10 hours of testing. The testing was performed on a single core running at 3.00 GHz with access to 4.0 MB of RAM. The operating system used was Windows 7, and the code was compiled using the GNU C++ compiler version 4.5.2 in the Netbeans 6.9.1 IDE[32]. Every 5 minutes, a timestamp, number of colorings found, and the current coloring under consideration (to identify how far through the search the algorithm had progressed) were logged to a file. Here is an abbreviated chart of the data collected, highlighting "round" values of time and number of colorings found for comparison's purposes:

Time Elapsed (Minutes)	Number of Colorings Found
60	40,021,843
120	80,411,355
155	101,426,216
180	116,741,823
250	155,178,294
370	213,753,643
375	217,422,059
555	368,217,284
600	376,204,182

At the 600 minute mark, the coloring currently being considered by the algorithm was:

00000111
01111000
01222122
01233213
01323231
12301120
13310321
13103200

Converting to a base-10 number, we have approximately 1.09×10^{35} out of a total space of $4^{8\times8} = 4^{64} \simeq 3.4 \times 10^{38}$ possible colorings. This gives a progress of approximately 0.03205% through the total search. Therefore, we can expect to search to continue approximately 3120.12 times longer than it already has before it completes. Given that the search took 10 hours to make this progress, that gives an expected total runtime of 31,201.2 processor-hours. In other words, if run on a 1,000-core supercomputer, the entire first phase should take approximately 31 hours to complete.

In terms of memory requirements, as mentioned before the RAM requirements of the first phase are inconsequential. However, the amount of data being generated is a serious concern. In testing, the program generated 376,204,182 colorings in 10 hours of execution on a single core. These colorings were stored in a series of files containing 50,000 colorings each. Altogether, there were 7,525 such files each requiring 3,565 KB of memory to store (except the last, which was not completely full). If this trend holds, that means every 10 hours of execution, there would be $7,545 \times 3,565$ KB = 26,897,925 KB $\times \frac{1MB}{1024KB} \times \frac{1GB}{1024MB} = 25.65$ GB of data generated. Over the entire first phase, that gives 25.65 GB $\times$ 3,120.12 = 80,036.88 GB $\times \frac{1TB}{1024GB} = 78$ TB of data.

Clearly 78 TB of data far exceeds the capabilities of all but the most specialized supercomputers, considering that we want to load these colorings into RAM eventually. Therefore, we provide two possible methods of thinning the candidate

quadrant-colorings of $G_{16,16}$. The first is to simply remove any colorings that are "identical" in the sense that entire colors are simply replaced by one another. For instance, given any legal coloring of $G_{8,8}$, we remove any coloring that simply swaps every color a with another color b and vice versa. This method would immediately lower the memory requirements to store the colorings of $G_{8,8}$ to $\frac{78TB}{4!} = 3.25$ TB. This, fortunately, is within range of the RAM capabilities of multiple supercomputers, as will be described in detail below.

One danger in using this method blindly is that it could be possible that a legal coloring of $G_{16,16}$ actually contains two or more "permuted" colorings, all but one of which would have been eliminated by this thinning process. However, each such coloring could later be reconstructed from the remaining coloring without a significant loss in runtime over simply reading and using the coloring itself. In particular, it would be possible to simply *check* if any permutations of a given coloring succeed without performing a single write or read operation (instead using modular arithmetic on the spot to simulate each such coloring).

The second method would be to make use of the constraints on the potential, legal 4-coloring of $G_{17,17}$ identified by Beth Kupin [28]. She discovered two and only two partial 1-colorings of $G_{17,17}$ that must be contained inside the final, legal $G_{17,17}$ 4-coloring, if it exists, up to the choice of permutation of the rows and columns. They are shown in Fig. 3.1.

In the figure, each symbol represents the assignment of the first color to a vertex of $G_{17,17}$ (note that the chart has dimension 17 by 17). The letters represent colorings of vertices that the two partial 1-colorings share in common, while ✠ and © are colorings of vertices that differ between the two 1-colorings. In fact however, a quick observation will show that these partial colorings are row-column permutations of one another. Swapping the rows containing C, D, E, and F with the rows containing G, H, I, and J and then swapping the first and second columns transforms one partial

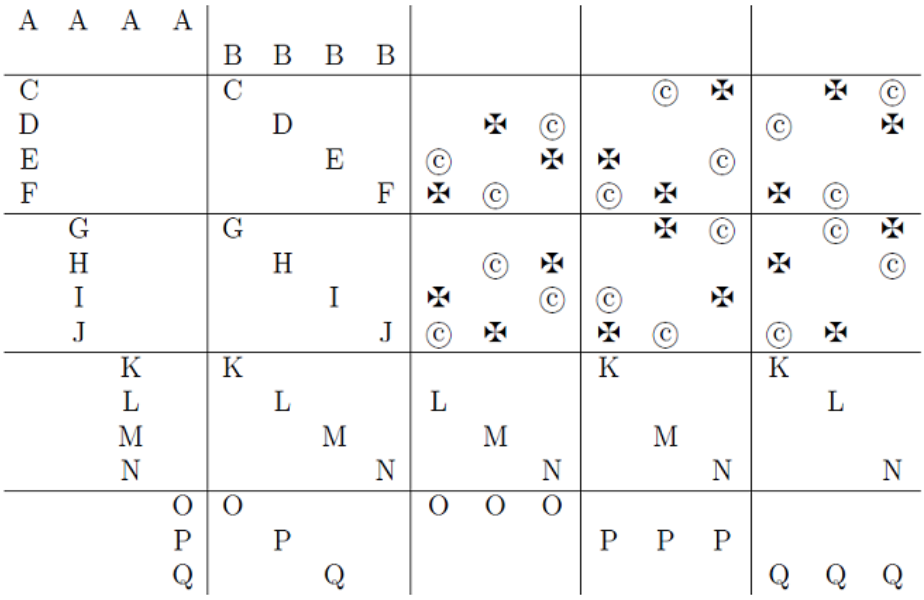

Figure 3.1: Kupin's two unique partial 1-colorings of $G_{17,17}$

coloring into the other, so with respect to a choice of row-column permutation, there must actually be *one* unique partial 1-coloring in the legal 4-coloring of $G_{17,17}$. [4] (Note that permuting rows and columns in any fashion preserves the legality or illegality of any monochromatic-rectangle-free coloring, since the vertices of the constituent rectangle constraints in each row or column are permuted in an identical fashion as the colorings themselves.)

Therefore, the second method would involve breaking the search for legal 4-colorings of $G_{8,8}$ into four separate searches of legal, effectively-3-colorings of $G_{8,8}$ and fixing the fourth color in the input. That is, the four runs would be seeded with four different inputs respectively, and the **goToNext** and **backtrack** procedures would simply "skip" over any occurences of the fourth color. The four inputs would correspond to the upper-left, upper-right, bottom-left, and bottom-right quadrants of the partial 1-coloring, excluding the far right row and far bottom column.

[4] We note that this observation also appeared in the comments of a complexity blog before publication of this thesis, here:
http://www.blogger.com/comment.g?blogID=3722233&postID=2725600741162296905

Altogether then, we find it reasonable to assume worst-case bounds of 31,201 processor-hours and 3.25 TB of output data representing at most approximately $376,204,182 \times 3,120.12 = 1.17 \times 10^{12}$ legal 4-colorings of $G_{8,8}$ out of the $4^{8\times 8} = 4^{64} > 10^{38}$ possible colorings of $G_{8,8}$ without using any of Kupin's results. Further, an approximate $\frac{10^{12}}{10^{38}} = 10^{-26}$ fraction of the possible colorings are legal colorings. Note the conservative rounding used; a more precise calculation would only give a more favorable estimate, in the context that follows.

In order to give *some* estimate of the number of legal colorings of $G_{8,8}$ by applying Kupin's partial 1-coloring of $G_{17,17}$, we first calculate the number of remaining, possible coloring extensions after seeding the search space with one of the resultant partial 1-colorings of $G_{8,8}$. In particular, ignoring the far-right column and far-bottom row of $G_{17,17}$, we dissect the space into four $G_{8,8}$ quadrants, and count the number of pre-colored cells in each quadrant via Fig. 3.1. In the upper-left, there are 20 pre-colored cells; in the upper-right, there are 22 pre-colored cells; in the bottom-left, there are 16 pre-colored cells; and in the bottom-right, there are 24 pre-colored cells. Thus, in the worst case, we are looking for all valid effectively-3-colorings of $G_{8,8}$ with 16 cells already having a fixed, fourth color. Therefore, there are $3^{8\times 8-16} = 3^{48} < 10^{23}$ possible colorings.

Unfortunately, there is no way to directly relate the fraction of legal 4-colorings of $G_{8,8}$ with the fraction of legal effectively-3-colorings of $G_{8,8}$ with 16 pre-colored cells. Since those 16 cells are guaranteed to be valid positions for the fourth color, the number of incorrect colorings will be reduced dramatically (i.e. we implicitly remove all of the illegal colorings over those cells from the search space). Indeed, a blind application of the fraction of legal 4-colorings of $G_{8,8}$ per possible colorings gives a nonsensical result: that there would be an estimated $10^{-26} \times 10^{23} = 10^{-3} < 1$ legal effectively-3-colorings of $G_{8,8}$. Instead, we do not try to estimate this value exactly, but simply state that the final number of 4-colorings of $G_{8,8}$ seeded with the four

partial colorings is likely to be significantly less (e.g., at least an order of magnitude or more) than the number of legal 4-colorings of $G_{8,8}$ without any pre-colored cells. For the purposes of the next section, let the maximum number of such colorings over each of the partial 1-colorings be $\psi_1^4(G_{8,8})$, given that $1 \leq \psi_1^4(G_{8,8}) < 1.17 \times 10^{12}$.

Finally, we point out that the execution of Phase 1 could be performed locally on the University of Arkansas RAZOR cluster[12]. In particular, RAZOR is a 1,512-core cluster with each core having a clock speed of 2.93 GHz with 27 TB of shared scratch storage. Utilizing the entire system, Phase 1 would likely be completed in a day on RAZOR with the runtime extending proportionally longer as less of the system is allocated toward this one computation.

3.3.2 Phase 2

For the second phase, we will use a modified backtracking approach to enumerate all legal, monochromatic-rectangle-free 4-colorings of $G_{16,16}$. In particular, we will treat every legal 4-coloring of $G_{8,8}$ as a "colored tile" of the appropriate quadrant of $G_{16,16}$. This, in essence, transforms the problem in the worst case into a search for all legal, "monochromatic-rectangle-free" $\psi_4^1(G_{8,8})$-colorings of $G_{2,2}$, where the coloring tiles of $G_{2,2}$ conflict (i.e., cause a candidate coloring to be illegal) *iff* a monochromatic rectangle is formed by the constituent colored cells of each tile.

To simulate the typical execution of this phase, we implemented a serial version of the program and ran the code for one hour with universe sizes of 50,000 and 100,000 coloring tiles (and for a handful of minutes with 200,000 coloring tiles) using the same testbed as with Phase 1. The key difficulty associated with a simulation of this phase is that its input data is dependent on the results of Phase 1, which we have not run in its entirety due to administrative and time constraints. Thus, to approximate the execution of Phase 2 as closely as possible, we used actual data (legal 4-colorings of $G_{8,8}$ arising from the appropriate partial 1-colorings) generated by the testing done

for Phase 1. Here are charts of the data collected:

Universe Size: 50,000 coloring tiles		
Time Elapsed (Seconds)	Progress	Projected Total Time Required
30.015	529 / 50,000	.79 hours
300.146	5327 / 50,000	.78 hours
600.292	10648 / 50,000	.78 hours
900.436	15961 / 50,000	.78 hours
1200.411	21297 / 50,000	.78 hours
2401.12	43600 / 50,000	.76 hours
2744.1	50,000 / 50,000	.76 hours

Universe Size: 100,000 coloring tiles		
Time Elapsed (Seconds)	Progress	Projected Total Time Required
30.014	261 / 100,000	3.33 hours
300.146	2613 / 100,000	3.19 hours
600.292	5258 / 100,000	3.17 hours
900.437	7946 / 100,000	3.15 hours
1200.58	10569 / 100,000	3.16 hours
2401.17	21532 / 100,000	3.1 hours
3601.75	32442 / 100,000	3.08 hours

Universe Size: 200,000 coloring tiles		
Time Elapsed (Seconds)	Progress	Projected Total Time Required
30.015	133 / 200,000	12.54 hours
60.029	268 / 200,000	12.44 hours
120.059	538 / 200,000	12.4 hours
300.146	1349 / 200,000	12.36 hours

Then, as a rough extrapolation of total time required to complete the search as the universe size grows, we can estimate that the total runtime increases fourfold everytime the universe size doubles (note that this is intended only to capture the time required up to an order of magnitude, and should be read as such):

Estimated Phase 2 Runtimes	
Universe Size	Projected Total Time Required
400,000	2 days
800,000	8 days
1.6×10^6	32 days
3.2×10^6	128 days
6.4×10^6	512 days
1.28×10^7	2048 days
1.024×10^8	16384 days
1.6384×10^9	262144 days
1.31072×10^{10}	2097152 days
1.048576×10^{11}	16777216 days
1.6777216×10^{12}	268435456 days

Fortunately, these timings assume that Phase 2 is being run on a single 3.00 GHz processor. If the search is instead run on a 1,000-core supercomputer (with a sufficiently large RAM, in order to store all of the 3.25 TB of coloring tiles in memory simultaneously), the process is sped up considerably. In particular, if we assume that $\psi_1^4(G_{8,8}) \simeq 10^9$ or less and given access to a 1,000-core high-RAM cluster, the total runtime of Phase 2 would take approximately $\frac{262144 \text{ days}}{1000} = 262.144 \text{ days} < 1 \text{ year}$.

For an example of such a system, we propose the use of the Blacklight supercomputer at the Pittsburgh Supercomputing Center (PSC) [34]. Blacklight is a large-scale shared-memory system hosting 4096 cores with a clock rate of 2.27 GHz with a total RAM capacity of 32 TB. It is further worthwhile to note that there are good oppor-

tunities for gaining access to such a system. In fact, as of the time of the writing of this thesis, Blacklight's administrators were actively looking for additional users with jobs that fundamentally required a high-RAM solution[42].

Finally, after the execution of Phase 2, we would have some number of legal 4-colorings of $G_{16,16}$. Let this number be $\psi_0^4(G_{16,16})$. In fact, $\psi_0^4(G_{16,16})$ *should* be quite small, since $G_{16,16}$ is known to be close to OBS_4. We then seed a brute-force (or backtracking-based) search for legal 4-colorings of $G_{17,17}$ with each of the legal 4-colorings of $G_{16,16}$. Each of the total, legal 4-colorings of $G_{16,16}$ require 33 more cells to be colored to complete a 4-coloring of $G_{17,17}$, which gives a search space of $\psi_0^4(G_{16,16}) \cdot 4^{33} < \psi_0^4(G_{16,16}) \cdot 10^{20}$. For values of $\psi_0^4(G_{16,16}) < 1000$ (which we see as overwhelmingly likely), this search is on the order of the Phase 1 search or less, which could be completed in one day.

3.4 Questions for Further Research

- What is the order of magnitude of $\phi_1^4(G_{8,8})$? By all accounts, this is a relatively simple computation on a supercomputer that we were unable to complete due to time constraints and other issues. We expect it to be on the order of 10^9, but it could be either more or less.

- Moreover, we would like to implement the computational plan described in this chapter in order to get an actual answer for the 17x17 4-Color Problem. Completing Stage 1 is a good first goal, but generating parallelizable code for a cluster for a long-term computation is an important next step as well. Also, are there any better high-performance computing techniques that can be applied? One interesting area for future exploration would be to design code to run on GPUs in order to speed up the computation even further. The problem has many symmetries that GPU computation could naturally exploit.

- Is there a more efficient algorithm for Monochromatic-Rectangle-Free Grid Coloring (or generally, any problem in Shape-Free Grid Coloring)? There are exciting theoretical and practical motivations for finding a subexponential time algorithm for the problem. One area for possible progress, as just mentioned, would be to leverage the natural symmetries in grid coloring to gain a theoretical speedup as well. For instance, see [4].

Bibliography

[1] Aaronson, Kuperberg, and Granade. Complexity Zoo. Qwiki. http://qwiki.stanford.edu/index.php/Complexity_Zoo

[2] Abrahamson, Downey, and Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogs. Annals of Pure and Applied Logic, 73, pages 235-276, 1995.

[3] Adler. Marshals, Monotone Marshals, and Hypertree-Width. Journal of Graph Theory, 47, pages 275-296, 2004.

[4] Aloul, Ramani, Markov, and Sakallah. Solving Difficult SAT Instances in the Presence of Symmetry. In Proceedings of the 39th Annual ACM Design Automation Conference, pages 731-736, 2002.

[5] Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In AFIPS spring joint computer conference, 1967.

[6] Apon. Grid k-coloring without monochromatic rectangles. Theoretical Computer Science Stack Exchange. http://cstheory.stackexchange.com/questions/791/grid-k-coloring-without-monochromatic-rectangles

[7] Apon. CSPs with unbounded fractional hypertree width. Theoretical Computer Science Stack Exchange. http://cstheory.stackexchange.com/questions/2008/csps-with-unbounded-fractional-hypertree-width

[8] Arora and Barak. Computational Complexity: A Modern Approach. 2009.

[9] Art of Problem Solving. http://www.artofproblemsolving.com/Forum/viewtopic.php?f=592&t=290906

[10] Bodirsky and Grohe. Non-dichotomies in Constraint Satisfaction Complexity. In Proceedings of the 35th Annual International Colloquium on Automata, Languages, and Programming, pages 184-196, 2008.

[11] Chandra, Furst, and Lipton. Multi-Party Protocols. In Proceedings of the 15th Annual ACM Symposium on Theory of Computing, pages 94-99, 1983.

[12] CI-TRAIN. Cyberinfrastructure for Transformational Scientific Discovery. Razor Cluster. http://www.ci-train.org/resources/equipment/computing.html

[13] Fenner, Gasarch, Glover, and Purewal. Rectangle Free Coloring of Grids. 2010. To appear.

[14] Garey and Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. 1979.

[15] Gasarch. Computational Complexity Blog. Scientific American Partner Network. http://blog.computationalcomplexity.org/2009/11/17x17-challenge-worth-28900-this-is-not.htm

[16] Gasarch. Computational Complexity Blog. Scientific American Partner Network. http://blog.computationalcomplexity.org/2009/12/17x17-comments-on-your-comments.html

[17] GNU project. GCC 4.5 Release Series. http://gcc.gnu.org/gcc-4.5/

[18] Grohe. The Complexity of Homomorphism and Constraint Satisfaction Problems Seen from the Other Side. In Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, pages 552-561, 2003.

[19] Grohe, Schwentick, and Segoufin. When is the Evaluation of Conjunctive Queries Tractable. In Proceedings of the 33rd ACM Symposium on Theory of Computing, pages 657-666, 2001.

[20] Grohe and Marx. Constraint Solving via Fractional Edge Covers. In Proceedings of the 17th Annual ACM Symposium on Discrete Algorithms, pages 289-298, 2006.

[21] Gross and Yellen. Graph Theory and Its Applications. Second Edition, 2006.

[22] Håstad. Some optimal inapproximability results. In Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pages 1-10, 1997.

[23] Hayes. bit-player. http://bit-player.org/2009/17-x-17-a-nonprogress-report

[24] Hayes. bit-player. http://bit-player.org/2010/whack-a-rectangle

[25] Impagliazzo, Paturi, and Zane. Which Problems have Strongly Exponential Complexity? Journal of Computer and System Sciences, 63 (4), pages 512-530, 2001.

[26] Kempton. ideas Revolution. http://kempton.wordpress.com/2009/11/30/two-prizes/

[27] Kirkpatrick, Gelatt, and Vecchi. Optimization by Simulated Annealing. In Science, 220 (4598), 1983.

[28] Kupin. Notes on 4-coloring the 17 by 17 grid. Unpublished manuscript, 2009. Available at http://www.cs.umd.edu/∼gasarch/BLOGPAPERS/bethk.pdf

[29] Kushilevitz and Nisan. Communication Complexity. 1997.

[30] Ladner. On the Structure of Polynomial Time Reducibility. Journal of the ACM, 22, 1, pages 155-171, 1975.

[31] Mahaney. Sparse complete sets for NP: Solutions of a conjecture by Berman and Hartmanis. Journal of Computer and System Sciences, 25, pages 130-143, 1982.

[32] Oracle Corporation. Netbeans, version 6.9. http://netbeans.org/community/releases/69/

[33] Papadimitriou. Computational Complexity. 1995.

[34] Pittsburgh Supercomputing Center. Blacklight. http://www.psc.edu/machines/sgi/uv/blacklight.php

[35] Puttgunta.17×17 almost coloring. http://www.cs.umd.edu/∼gasarch/BLOGPAPERS/17x17almost.txt

[36] Ramsey. On a Problem of Formal Logic. In Proceedings of the London Mathematical Society, s2-30, pages 264-286, 1930.

[37] Schweitzer. HTML5 Rectangle-Free-Coloring Game. http://www.martinschweitzer.com/squaregame.html

[38] Seymour and Thomas. Graph Searching, and a Min-Max Theorem for Tree-Width. Journal of Combinatorial Theory, Series B, 58, pages 22-33, 1993.

[39] Thiery. 17×17 Challenge. http://linbaba.wordpress.com/17x17-challenge/

[40] Van der Waerden. Beweis einer Baudetschen Vermutung. Nieuw Archief voor Wiskunde, 15, pages 212-216, 1927.

[41] William Gasarch (UMD-College Park), Jim Purtilo (UMD-College Park). Personal communication, Spring 2011.

[42] Jeff Pummill (UofA), Ralph Roskies (PSC). Personal communication, Spring 2011.