The Complexity of Grid Coloring

Daniel Apon [*]
Univ. of MD at College Park

William Gasarch [†]
Univ. of MD at College Park

Kevin Lawler [‡]
Permanent

September 14, 2021

Abstract

A c-coloring of the grid $G_{N,M} = [N] \times [M]$ is a mapping of $G_{N,M}$ into $[c]$ such that no four corners forming a rectangle have the same color. In 2009 a challenge was proposed via the internet to find a 4-coloring $[17] \times [17]$. Though a coloring was produced, finding it proved to be difficult. We present three results that indicate that finding grid colorings is hard. (1) Given a partial c-coloring of the $G_{N,M}$ grid, can it be extended to a full c-coloring? We show this problem is NP-complete. (2) The statement $G_{N,M}$ *is c-colorable* can be expressed as a Boolean formula GRID(n, m, c) with nmc variables. We show that if the $G_{N,M}$ is not c-colorable then any tree resolution proof that GRID(n, m, c) is not satisfiable is of size $2^{\Omega(c)}$. We then generalize this result for other monochromatic shapes. (3) We show that any tree-like cutting planes proof that $c+1$ by $c\binom{c+1}{2}+1$ is not c-colorable must be of size $2^{\Omega(c^3/\log^2 c)}$. Note that items (2) and (3) yield statements from Ramsey Theory which are of size polynomial in their parameters and require exponential size in various proof systems.

[*]University of Maryland,College Park, MD 20742. dapon.crypto@gmail.com
[†]University of Maryland, College Park, MD 20742. gasarch@cs.umd.edu
[‡]Permanent, Berkeley, CA 94710. kevin@permanentco.com

1 Introduction

Notation 1.1

1. If $x \in \mathbb{N}$ then $[x]$ denotes the set $\{1, \ldots, x\}$. $G_{N,M}$ is the set $[N] \times [M]$.
2. If X is a set and $k \in \mathbb{N}$ then $\binom{X}{k}$ is the set of all size-k subsets of X.

Def 1.2 A *rectangle* of $G_{N,M}$ is a subset of the form $\{(a,b),(a+c_1,b),(a+c_1,b+c_2),(a,b+c_2)\}$ for some $a,b,c_1,c_2 \in \mathbb{N}$. Note that we are only looking at the four corners of the rectangle—nothing else. A grid $G_{N,M}$ is *c-colorable* if there is a function $\chi : G_{N,M} \to [c]$ such that there are no rectangles with all four corners the same color. In other words, a grid $G_{N,M}$ is *c-colorable* if there is a function $\chi : G_{N,M} \to [c]$ such that there are no monochromatic rectangles.

Fenner et al. [2] explored the following problem:

Which grids are c-colorable for a given fixed c?

We state some of their results.

1. For all $c \geq 2$, $G_{c+1,\binom{c+1}{2}+1}$ is not c-colorable
2. For all c there exists a finite number of grids such that $G_{N,M}$ is c-colorable iff it fails to contain any one of those grids. This set of grids is called *the obstruction set* and is denoted by OBS_c.
3. $\mathrm{OBS}_2 = \{G_{3,7}, G_{5,5}, G_{7,3}\}$. This was obtained without the aid of a computer program.
4. $\mathrm{OBS}_3 = \{G_{19,4}, G_{16,5}, G_{13,7}, G_{11,10}, G_{10,11}, G_{7,13}, G_{5,16}, G_{4,19}\}$. A computer aided search was used to find a 3-coloring of $G_{10,10}$.
5. $$\mathrm{OBS}_4 = \{G_{41,5}, G_{31,6}, G_{29,7}, G_{25,9}, G_{23,10}, G_{22,11}, G_{21,13}, G_{19,17}\} \bigcup$$
$$\{G_{17,19}, G_{13,21}, G_{11,22}, G_{10,23}, G_{9,25}, G_{7,29}, G_{6,31}, G_{5,41}\}$$
The authors were stuck for a long time trying to find a 4-coloring of $G_{17,17}$. William Gasarch put a bounty of $17^2 = 289$ dollars for a

4-coloring of $G_{17,17}$ and posted this challenge to ComplexityBlog [3]. Bernd Steinbach and Christian Posthoff [12, 13, 14, 4] found a 4-coloring of $G_{17,17}$ and also of $G_{18,18}$, which completed the problem. They used a rather sophisticated SAT solver that was tuned to grid coloring problems.

6. Finding OBS_5 seems to be beyond current technology.

The difficulty of 4-coloring $G_{17,17}$ and pinning down OBS_5 raise the following question: is the problem of grid coloring hard? In this paper we formalize and prove three different results that indicate grid coloring is hard. We will also give a fixed parameter tractable approach which may make the problem more tractable.

1.1 Grid Coloring Extension is NP-Complete

Def 1.3 Let $N, M, c \in \mathbb{N}$.

1. A *partial mapping χ of $G_{N,M}$ to* $[c]$ is a mapping of a subset of $G_{N,M}$ to $[c]$. See Figure 1 for an example.

2. If χ is a partial mapping of $G_{N,M}$ to $[c]$ then χ' is *an extension of* χ if χ' is a partial mapping of $G_{N,M}$ to $[c]$ which (1) is defined on every point that χ is defined, (2) agrees with ϕ on those points, and (3) may be defined on more points .

3. A *partial mapping χ of $G_{N,M}$ to* $[c]$ is a mapping of $G_{N,M}$ to $[c]$. This would normally just be called a mapping, but we use the term total to distinguish it from a partial mapping.

Def 1.4 Let $c, N, M \in \mathbb{N}$. A partial coloring χ of $G_{N,M}$ to $[c]$ is *extendable to a c-coloring* if there is an extension of χ to a total mapping which is a c-coloring of $G_{N,M}$. We will use the term *extendable* if the c is understood.

Def 1.5 Let

$$\text{GCE} = \{(N, M, c, \chi) : \chi \text{ is extendable}\}.$$

GCE stands for *Grid Coloring Extension.*

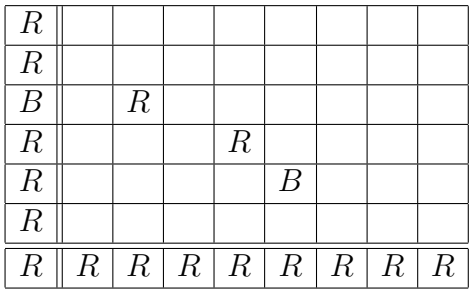

Figure 1: Example of a partial coloring

We show that GCE is NP-complete. This result may explain why the original 17×17 challenge was so difficult. Then again—it may not. GCE with parameter c is Fixed Parameter Tractable.

There is another reason the results obtained may not be the reason why the 17×17 challenge was hard. The 17×17 challenge can be rephrased as proving that $(17, 17, 4, \chi) \in$ GCE where χ is the empty partial coloring. This is a special case of GCE since none of the spots are pre-colored. It is possible that the case where χ is the empty coloring is easy. While we doubt this is true, we note that we have not eliminated the possibility.

One could ask about the problem

$$\text{GC} = \{(n, m, c) : G_{N,M} \text{ is } c\text{-colorable } \}.$$

However, if n, m are in unary, then GC is a sparse set. By Mahaney's Theorem [9, **?**] if a sparse set is NP-complete then P = NP. If n, m are in binary, then we cannot show that GC is in NP since the obvious witness is exponential in the input. This formulation does not get at the heart of the problem, since we believe it is hard because the number of possible colorings is large, not because n, m are large. It is an open problem to find a framework within which a problem like GC can be shown to be hard.

2 GCE is NP-complete

Theorem 2.1 GCE *is* NP*-complete.*

Proof:

Clearly GCE $\in$ NP. Let $\phi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$ be a 3-CNF

formula with n free variables and m clauses. We determine N, M, c and a partial c-coloring χ of $G_{N,M}$ such that

$$\phi \in \text{3-SAT iff } (N, M, c, \chi) \in \text{GCE}.$$

We visualize the full grid as a core subgrid with additional entries to the left and below. These additional entries are there to enforce that some colors in the core grid occur only once. Clauses will correspond to groupings of columns inside the core grid, and free variables to groupings of rows.

The colors will be T, F, and some of the $(i, j) \in G_{N,M}$. We use (i, j) to denote a color for a particular position.

We label the colors using the literals T, F and values of $(i, j) \in G_{N,M}$. Syntactically T, F are different from all of the (i, j).

The construction is in four parts. We summarize the four parts here before going into details.

Part One

We will often set $\chi(i, j)$ to (i, j) and then never reuse (i, j) in the core grid. By doing this, we make having a monochromatic rectangle rare and have control over when that happens.

We show how to color the cells that are not in the core grid to achieve this. We show this final step of the construction first.

Part Two

The core grid will have $2nm + 1$ rows. In the first column we have $2nm$ blank spaces and the space $(1, 2nm+1)$ colored with $(1, 2nm+1)$. The $2nm$ blank spaces will be forced to be colored T or F. We think of the column as being in n blocks of $2m$ spaces each. In the ith block the coloring will be forced as follows:

1. If x_i is to be set to F then the column is colored:

$$\begin{array}{c} T \\ F \\ \vdots \\ T \\ F \end{array}$$

$(2,4)$								
$(2,4)$								
$(2,4)$								
$(2,4)$								
T		$(2,4)$						
$(2,4)$								
$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$

Figure 2: Cell $(2,4)$ is colored $(2,4)$ and nothing else can be

2. if x_i is to be set to T then the column is colored:

$$\begin{matrix} F \\ T \\ \vdots \\ F \\ T \end{matrix}$$

Part Three

For each clause C there will be two columns. The coloring χ will be defined on most of the cells in these columns. However, the coloring will extend to these two columns iff one of the literals in C is colored T in the first column.

Part Four

We set the number of colors properly so that the T and F will be forced to be used in all blank spaces.

We now go into detail.

Part One: Forcing a color to appear only once in the core grid.

Say we want the cell $(2,4)$ in the core grid to be colored $(2,4)$ and we do not want this color appearing anywhere else in the core grid. We can do the following: add a column of $(2,4)$'s to the left end (with one exception) and a row of $(2,4)$'s below. See Figure 2. In this can all later figures the left bottom cell of the core grid is indexed $(1,1)$.

(The double lines are not part of the construction. They are there to separate the core grid from the rest.)

It is easy to see that in any coloring of the above grid the only cells that can have the color $(2,4)$ are those shown to already have that color.

$(5,3)$	$(2,4)$								
$(5,3)$	$(2,4)$								
$(5,3)$	T		$(2,4)$						
T	$(2,4)$					$(5,3)$			
$(5,3)$	$(2,4)$								
$(5,3)$	$(2,4)$								
$(5,3)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$	$(2,4)$
$(5,3)$	$(5,3)$	$(5,3)$	$(5,3)$	$(5,3)$	$(5,3)$	$(5,3)$	$(5,3)$	$(5,3)$	$(5,3)$

Figure 3: $(2,4)$ and $(5,3)$

It is also easy to see that the color T we have will not help to create any monochromatic rectangles since there are no other T's in its column. The T we are using *is* the same T that will later mean *true*. We could have used F. We do not want to use new colors since we would have no control over where else they could be used.

What if some other cell needs to have a unique color? Let's say we also want to color cell $(5,3)$ in the core grid with $(5,3)$ and do not want to color anything else in the core grid $(5,3)$. Then we use the grid in Figure 3

It is easy to see that in any coloring of the above grid the only cells that can have the color $(2,4)$ or $(5,3)$ are those shown to already have those colors.

For the rest of the construction we will only show the core grid. If we denote a color as D (short for *Distinct*) in the cell (i,j) then this means that (1) cell (i,j) is color (i,j) and (2) we have used the above gadget to make sure that (i,j) does not occur as a color in any other cell of the core grid. Note that when we have D in the $(2,4)$ cell and in the $(5,3)$ cell, they denote different colors.

Part Two: Forcing $(x,\overline{x})$ to be colored (T,F) or (F,T).

There will be one column with cells labeled by literals. The cells are uncolored. We will call this row *the literal column*. We will put to the left of the literal column, separated by a triple line, the literals whose values we intend to set. These literals are not part of the construction; they are a visual aid. The color of the literal-labeled cells will be T or F. We need to make sure that all of the x_i have the same color and that the color is different than that of $\overline{x}_i$.

$\overline{x}_1$		T	F
x_1		T	F

Figure 4: Forcing x_1 and $\overline{x}_1$

	D	D	D	D	D	D	D	D	D	D	D
$\overline{x}_1$		D	D	D	D	D	D	D	D	T	F
x_1		D	D	D	D	D	D	T	F	T	F
$\overline{x}_1$		D	D	D	D	T	F	T	F	D	D
x_1		D	D	T	F	T	F	D	D	D	D
$\overline{x}_1$		T	F	T	F	D	D	D	D	D	D
x_1		T	F	D	D	D	D	D	D	D	D

Figure 5: Forcing when need three copies of x_1 or $\overline{x}_1$

Figure 4 is an example which shows how we can force $(x_1, \overline{x}_1)$ to be colored (T, F) or (F, T).

We will actually need (at most) m copies of x_1 and m copies of $\overline{x}_1$. We will also put a row of D's on top which we will use later. Figure 2 illustrates how to do this when you need 3 copies of x_1 or $\overline{x}_1$.

We leave it as an exercise to prove that

- If the bottom x_1 cell is colored T then (1) all of the x_1 cells are colored T, and (2) all of the $\overline{x}_1$ cells are colored F.
- If the bottom x_1 cell is colored F then (1) all of the x_1 cells are colored F, and (2) all of the $\overline{x}_1$ cells are colored T.

Note that (1) if we want one literal-pair (that is $x_1, \overline{x}_1$) then we use two columns, (2) if we want two literal-pairs then we use six columns, and (3) if we want three literal-pairs then we use ten columns. We leave it as an exercise to generalize the construction to m literal-pairs using $2 + 4(m-1)$ columns.

We will need m copies of x_1, $\overline{x}_1$, x_2 and $\overline{x}_2$. We illustrate how do to this in Figure 6. We use double lines in the picture to clarify that the x_1 and the x_2 variables are not chained together in any way.

We leave it as an exercise to prove that, for all $i \in \{1, 2\}$:

	D	D	D	D	D	D	D	D	D	D	D	D	D
$\overline{x}_2$		D	D	D	D	D	D	D	D	D	D	T	F
x_2		D	D	D	D	D	D	D	D	T	F	T	F
$\overline{x}_2$		D	D	D	D	D	D	T	F	T	F	D	D
x_2		D	D	D	D	D	D	T	F	D	D	D	D
$\overline{x}_1$		D	D	D	D	T	F	D	D	D	D	D	D
x_1		D	D	T	F	T	F	D	D	D	D	D	D
$\overline{x}_1$		T	F	T	F	D	D	D	D	D	D	D	D
x_1		T	F	D	D	D	D	D	D	D	D	D	D

Figure 6: Two x_1's, $\overline{x}_1$'s, x_2, and $\overline{x}_2$

- If the bottom x_i cell is colored T then (1) all of the x_i cells are colored T, and (2) all of the $\overline{x}_1$ cells are colored F.
- If the bottom x_i cell is colored F then (1) all of the x_i cells are colored F, and (2) all of the $\overline{x}_1$ cells are colored T.

An easy exercise for the reader is to generalize the above to a construction with n variables with m literal-pairs for each variable. This will take $n(2 + 4(m-1))$ columns.

For the rest of the construction we will only show the literal column and the clause columns (which we define in the next part). It will be assumed that the D's and T's and F's are in place to ensure that all of the x_i cells are one of $\{T, F\}$, and the $\overline{x}_i$ cells are the other color.

Part Three: How we force the coloring to satisfy a single clause

Say one of the clauses is $C_1 = L_1 \vee L_2 \vee L_3$ where L_1, L_2, and L_3 are literals. Pick an L_1 row, an L_2 row, and an L_3 row. We will also use the top row, as we will see. For other clauses you will pick other rows. Since there are m copies of each variable and its negation, this is easy to do.

The two T's in the top row in the next picture are actually in the very top row of the grid.

We put a C_1 over the columns that will enforce that C_1 is satisfied. We put L_1, L_2, and L_3 on the side to indicate the positions of the variables. These C_1 and the L_i outside the triple bars are not part of the grid. They are a visual aid.

This is the partially colored grid used for the clause $L_1 \vee L_2 \vee L_3$.

		C_1	C_1
	D	T	T
L_3		D	F
L_2			
L_1		F	D

Claim 1: If χ' is a 2-coloring of the blank spots in this grid (with colors T and F) then it cannot have the L_1, L_2, L_3 spots all colored F.

Proof of Claim 1:

Assume, by way of contradiction, that that L_1, L_2, L_3 are all colored F. Then this is what it looks like:

		C_1	C_1
	D	T	T
L_3	F	D	F
L_2	F		
L_1	F	F	D

The two blank spaces are both forced to be T since otherwise you get a monochromatic rectangle of color F. Hence we have

		C_1	C_1
	D	T	T
L_3	F	D	F
L_2	F	T	T
L_1	F	F	D

This coloring has a monochromatic rectangle which is colored T. This contradicts χ' being a 2-coloring of the blank spots.

End of Proof of Claim 1

We leave the proof of Claim 2 below to the reader.

Claim 2: If χ' colors L_1, L_2, L_3 anything except F, F, F then χ' can be extended to a coloring of the grid shown.

Upshot: A 2-coloring of the grid is equivalent to a satisfying assignment of the clause.

Note that each clause will require 2 columns to deal with. So there will be $2m$ columns for this.

Part Four: Putting it all together

Recall that $\phi(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$ is a 3-CNF formula. We first define the core grid and later define the entire grid and N, M, c. The core grid will have $2nm + 1$ rows and $n(4m - 2) + 2m + 1$ columns. The first $n(4m-2)+1$ columns are partially colored using the construction in Part 2. This will establish the literal column. We will later set the number of colors so that the literal column must use the colors T and F.

For each of the m clauses we pick a set of its literals from the literals column. These sets-of-literals are all disjoint. We can do this since we have m copies of each literal-pair. We then do the construction in Part 3. Note that this uses two columns. Assuming that all of the D's are colored distinctly and that the only colors left are T and F, this will ensure that the core grid is c-colorable iff the formula is satisfiable.

The core grid is now complete. For every (i, j) that is colored (i, j), we perform the method in Part 1 to make sure that (i, j) is the only cell with color (i, j). Let the number of such (i, j) be C. The number of colors c is $C + 2$. ∎

3 An Example

We can make the construction slightly more efficient (and thus can actually work out an example). We took m pairs $\{x_i, \overline{x}_i\}$. We don't really need all m. If x_i appears in a clauses and $\overline{x}_i$ appears in b clauses then we only need $\max\{a, b\}$ literal-pairs. If $a \neq b$ then we only need $\max\{a, b\} - 1$ literal-pairs and one additional literal. (This will be the case in the example below.)

With this in mind, we will do an example—though we will only show the core grid.

$$(x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_2 \vee x_3 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4)$$

We only need

- one $(x_1, \overline{x}_1)$ literal-pair,
- one $(x_2, \overline{x}_2)$ literal-pair,
- one $(x_3, \overline{x}_3)$ literal-pair,
- one additional $\overline{x}_3$,

												C_1	C_1	C_2	C_2	C_3	C_3
	D	D	D	D	D	D	D	D	D	D	D	T	T	T	T	T	T
$\overline{x}_4$		D	D	D	D	D	D	D	D	T	F	D	D	D	D	D	F
x_4		D	D	D	D	D	D	D	D	T	F	D	D	D	F	D	D
$\overline{x}_3$		D	D	D	D	D	D	T	F	D	D	D	D	D	D	D	D
x_3		D	D	D	D	T	F	T	F	D	D	D	D			D	D
$\overline{x}_3$		D	D	D	D	T	F	D	D	D	D	D	F	D	D		
$\overline{x}_2$		D	D	T	F	D	D	D	D	D	D	D	D	F	D	D	D
x_2		D	D	T	F	D	D	D	D	D	D			D	D	D	D
$\overline{x}_1$		T	F	D	D	D	D	D	D	D	D	D	D	D	D	F	D
x_1		T	F	D	D	D	D	D	D	D	D	F	D	D	D	D	D

Figure 7: Example with $(x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_2 \vee x_3 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4)$

- one $(x_4, \overline{x}_4)$ literal-pair.

See Figure 3 for the core grid.

4 Fixed Parameter Tractability

The 17×17 problem only involved 4-colorability. Does the result that GCE is NP-complete really shed light on the hardness of the 17×17 problem?

Consider the problem where the number of colors is fixed at some c. We will see that this problem is Fixed Parameter Tractable.

Def 4.1 Let $c \in \mathbb{N}$. Let

$$\text{GCE}_c = \{(N, M, \chi) : \chi \text{ can be extended to a } c\text{-coloring of } G_{N,M} \ \}.$$

Clearly $\text{GCE}_c \in \text{DTIME}(c^{O(NM)})$. Can we do better? Yes. We will show that GCE is in time $O(N^2M^2 + 2^{O(c^4)})$.

Lemma 4.2 *Let n, m, c be such that $c \leq 2^{nm}$. Let χ be a partial c-coloring of $G_{n,m}$. Let U be the uncolored grid points. Let $|U| = u$. There is an algorithm that will determine if χ can be extended to a full c-coloring that runs in time $O(cnm2^{2u})$.*

Proof: For $S \subseteq U$ and $1 \leq i \leq c$ let

$$f(S,i) = \begin{cases} \text{YES} & \text{if } \chi \text{ can be extended to color } S \text{ using only colors } \{1,\ldots,i\}; \\ \text{NO} & \text{if not.} \end{cases} \tag{1}$$

We assume throughout that the coloring χ has already been applied.

We are interested in $f(U,c)$; however, we use a dynamic program to compute $f(S,i)$ for all $S \subseteq U$ and $1 \leq i \leq c$. Note that $f(\emptyset,i) = \text{YES}$.

We describe how to compute $f(S,i)$. Assume that for all S' such that $|S'| < |S|$, for all $1 \leq i \leq c$, $f(S',i)$ is known.

1. For all nonempty 1-colorable $T \subseteq S$ do the following (Note that there are at most 2^u sets T.)
   (a) If $f(S-T,i) = \text{NO}$ then $f(S,i) = \text{NO}$.
   (b) If $f(S-T,i-1) = \text{YES}$ then determine if coloring T with i will create a monochromatic rectangle. If not then $f(S,i) = \text{YES}$. Note that this takes $O(nm)$.
2. We now know that for all 1-colorable $T \subseteq S$ (1) $f(S-T,i) = \text{YES}$, and (2) either $f(S-T,i-1) = \text{NO}$ or $f(S-T,i-1) = \text{YES}$, and coloring T with i creates a monochromatic rectangle. We will show that in this case $f(S,i) = \text{NO}$.

Assume that, for all 1-colorable sets $T \subseteq S$: (1) $f(S-T,i) = \text{YES}$, and (2) either $f(S-T,i-1) = \text{NO}$ or $f(S-T,i-1) = \text{YES}$ and coloring T with i creates a rectangle with χ. Also assume, by way of contradiction, that $f(S,i) = \text{YES}$. Let COL be an extension of χ to S. Let T be the set colored i. Clearly $f(S-T,i-1) = \text{YES}$. Hence the second clause of condition (2) must hold. Hence coloring T with i creates a monochromatic rectangle. This contradicts COL being a c-coloring.

The dynamic program fills in a table that is indexed by the 2^u subsets of S and the c colors. Each slot in the table takes $O(nm2^u)$ to compute. Hence filling the entire table takes $O(cnm2^{2u})$ steps. ∎

The following result is due to Fenner et al. [2].

Lemma 4.3 *Assume $c+1 \leq N$ and $c\binom{c+1}{2} < M$. Then $G_{N,M}$ is not c-colorable. Hence, for any χ, $(N,M,\chi) \notin \text{GCE}_c$.*

Proof: Assume, by way of contradiction, that there is a c-coloring of $G_{N,M}$. Since every column has at least $c+1$ elements the following mapping is well defined: map every column to the least $(\{i,j\},a)$ such that the $\{i,j\} \in \binom{[c+1]}{2}$ and both the ith and the jth row of that column are colored a. The codomain of this function has more than $c\binom{c+1}{2}$ elements. Hence some element of the codomain is mapped to at least twice. This yields a monochromatic rectangle. ∎

Lemma 4.4 *Assume $N \leq c$ and $M \in \mathbb{N}$. If χ is a partial c-coloring of $G_{N,M}$ then $(N, M, \chi) \in \mathrm{GCE}_c$.*

Proof: The partial c-coloring χ can be extended to a full c-coloring as follows: for each column use a different color for each blank spot, making sure that all of the new colors in that column are different from each other. ∎

Theorem 4.5 $\mathrm{GCE}_c \in \mathrm{DTIME}(N^2M^2 + 2^{O(c^6)})$ *time.*

Proof:

1. Input (N, M, χ).
2. If $N \leq c$ or $M \leq c$ then test if χ is a partial c-coloring of $G_{N,M}$. If so then output YES. If not then output NO. (This works by Lemma 4.4.) This takes time $O(N^2M^2)$. Henceforth we assume $c+1 \leq N, M$.
3. If $c\binom{c+1}{2} < M$ or $c\binom{c+1}{2} < N$ then output NO and stop. (This works by Lemma 4.3.)
4. The only case left is $c+1 \leq N, M \leq c\binom{c+1}{2}$. By Lemma 4.2 we can determine if χ can be extended in time $O(2^{NM}) = O(2^{c^6})$.

Step 2 takes $O(N^2M^2)$, and Step 4 takes time $2^{O(c^6)})$. Hence the entire algorithm takes time $O(N^2M^2 + 2^{O(c^6)})$. ∎

Can we do better? Yes, but it will require a result from Fenner et al. [2].

Lemma 4.6 *Let $1 \leq c' \leq c-1$.*

1. *If $N \geq c + c'$ and $M > \frac{c}{c'}\binom{c+c'}{2}$ then $G_{N,M}$ is not c-colorable.*

2. *If $N \geq 2c$ and $M > 2\binom{2c}{2}$ then $G_{N,M}$ is not c-colorable. (This follows from a weak version of the $c' = c - 1$ case of Part 1.)*

Theorem 4.7 $\text{GCE}_c \in \text{DTIME}(N^2M^2 + 2^{O(c^4)})$ *time.*

Proof:

1. Input (N, M, χ).

2. If $N \leq c$ or $M \leq c$ then test if χ is a partial c-coloring of $G_{N,M}$. If so then output YES. If not then output NO. (This works by Lemma 4.4.) This takes time $O(N^2M^2)$.

3. For $1 \leq c' \leq c - 1$ we have the following pairs of cases.

   (a) If $N = c + c'$ and $M > \frac{c}{c'}\binom{c+c'}{2}$, then output NO and stop. (This works by Lemma 4.6.)

   (b) $N = c + c'$ and $M \leq \frac{c}{c'}\binom{c+c'}{2}$. By Lemma 4.2 we can determine if χ can be extended to a total c-coloring in time $2^{O(NM)}$. Note that $MN \leq (c+c')\frac{c}{c'}\binom{c+c'}{2}$. On the interval $1 \leq c' \leq c-1$ this function achieves its maximum when $c' = 1$. Hence this case takes $2^{O(c^4)}$.

   Henceforth we assume $2c \leq N, M$.

4. If $M > 2\binom{2c}{2}$ or $N > 2\binom{2c}{2}$ then output NO and stop. (This works by Lemma 4.6.)

5. The only case left is $2c \leq N, M \leq 2\binom{2c}{2}$. By Lemma 4.2 we can determine if χ can be extended in time $2^{O(NM)} \leq 2^{O(c^4)}$.

Step 2 and Step 4 together take time $O(N^2M^2 + 2^{O(c^4)})$. ∎

Even for small c the additive term $2^{O(c^4)}$ is the real timesink. A cleverer algorithm that reduces this term is desirable. By Theorem 2.1 this term cannot be made polynomial unless P=NP.

5 Lower Bound on Tree Res

For n, m, c we define a Boolean formula GRID(n, m, c) such that

$$G_{n,m} \text{ is } c\text{-colorable iff GRID}(n, m, c) \in \text{SAT}.$$

- The variables are x_{ijk} where $1 \leq i \leq n$, $1 \leq j \leq m$, $1 \leq k \leq c$. The intention is that, for all (i, j), there is a k such that x_{ijk} is true. We interpret k to be the color of (i, j).
- For all (i, j) we have the clause

$$\bigvee_{k=1}^{c} x_{ijk}.$$

  These clauses ensure that every (i, j) has at least one color.

- For all $1 \leq i < i' \leq n$ and $1 \leq j < j' \leq m$ we have the clause

$$\bigvee_{k=1}^{c} \neg x_{ijk} \vee \neg x_{i'jk} \vee \neg x_{ij'k} \vee \neg x_{i'j'k}.$$

  These clauses ensure there are no monochromatic rectangles.

We do not use clauses to ensure that every (i, j) has at most one color. This is because if the formula above is satisfied then one can extract out of it a c-coloring of $G_{n,m}$ by taking the color of (i, j) to be the *least* k such that x_{ijk} is true.

We show that if $G_{n,m}$ is not c-colorable then any tree resolution proof of GRID$(n, m, c) \notin$ SAT requires size $2^{\Omega(c)}$.

5.1 Background on Tree Resolution and the Prover-Delayer Game

The definitions of Resolution and Tree Resolution are standard; however, we define them for completeness. Prover-Delayer games were first defined by Pudlak and Impagliazzo [10], however we use the asymmetric version which was first defined by Beyersdorff et al. [1]. See also the book by Jukna [6].

Def 5.1 Let $\varphi = C_1 \wedge \cdots \wedge C_L$ be a CNF formula. A *Resolution Proof that* $\varphi \notin$ SAT is a sequence of clauses such that on each line you have one of the following:

1. One of the C's in φ (called an AXIOM).
2. $A \vee B$ where on prior lines you had $A \vee x$ and $B \vee \neg x$.
3. The last line has the empty clause.
4. For every resolution proof there is a graph: at the top there are nodes $C_1, \ldots, C_L$, and when step 2 above is applied you have edges from $A \vee x$ and $B \vee \neg x$ to $A \vee B$.

It is easy to see that if there is a resolution proof that $\varphi \notin$ SAT then indeed $\varphi \notin$ SAT. The converse is also true though slightly harder to prove.

Def 5.2 The *size of a resolution proof* is the number of nodes in the graph.

Def 5.3 A *Tree Resolution* proof is one whose graph is a tree.

Def 5.4 The *Prover-Delayer Game* has parameters (1) $a, b \in (1, \infty)$, such that $\frac{1}{a} + \frac{1}{b} = 1$, (2) $p \in \mathbb{R}^+$, and (3) a CNF-formula

$$\varphi = C_1 \wedge \cdots \wedge C_L \notin \text{SAT}.$$

The game is played as follows until a clause is proven false:

1. The Prover picks a variable x that was not already picked.
2. The Delayer either
   (a) Sets x to T or F.
   (b) Defers to the Prover.
      i. If the Prover sets x to F then the Delayer gets $\lg a$ points.
      ii. If the Prover sets x to T then the Delayer gets $\lg b$ points.

When some clause has all of its literals set to false the game ends. At that point, if the Delayer has p points then he wins; otherwise the Prover wins.

We assume that the Prover and the Delayer play perfectly.

1. *The Prover wins* means *the Prover has a winning strategy.*

2. *The Delayer wins* means *the Delayer has a winning strategy.*

Lemma 5.5 *Let $a, b \in (1, \infty)$ such that $\frac{1}{a} + \frac{1}{b} = 1$, $p \in \mathbb{R}^+$, $\varphi \notin$ SAT, φ in CNF-form. If the Delayer wins (by always getting $\geq p$ points) then every Tree Resolution proof for φ has size $\geq 2^p$.*

Note that the lower bound in Lemma 5.5 is 2^p, not $2^{\Omega(p)}$.

5.2 Lower Bound on Tree Resolution

Theorem 5.6 *Let n, m, c be such that $G_{n,m}$ is not c-colorable and $c \geq 9288$. Any tree resolution proof of* GRID$(n, m, c) \notin$ SAT *requires size 2^{Dc} where $D = 0.836$.*

Proof:

By Lemma 5.5 it will suffice to show that there exists $a, b \in (1, \infty)$ with $\frac{1}{a} + \frac{1}{b} = 1$, such that the Delayer wins the Prover-Delayer game with parameters a, b, Dc, and GRID(n, m, c). We will determine a, b later. We will also need parameter $r \in (0, 1)$ to be determined.

Here is the Delayers strategy: Assume x_{ijk} was chosen by the Prover.

1. If coloring (i, j) with color k will create a monochromatic rectangle then the Delayer will NOT let this happen—he will set x_{ijk} to F. The Delayer does not get any points but he avoids the game ending. (Formally: if there exists i', j' such that $x_{i'jk} = x_{ij'k} = x_{i'j'k} = T$ then the Delayer sets x_{ijk} to F.) Otherwise he goes to the next step of the strategy.

2. The Delayer is concerned that if none of the x_{ij*} are set to T and many are set to F then the game may be over soon. But he is reluctant to set x_{ijk} to T. Hence he will do it only if Prover has already set many vars to F, hence Delayer has already gotten lots of points. We now proceed formally. If there are at least rc values k' where the Prover has set $x_{ijk'}$ to F, and there are no $x_{ijk'}$ that have been set to T (by anyone) then Delayer sets x_{ijk} to T. Note that this cannot form a monochromatic rectangle since in step 1 of the strategy x_{ijk} would have been set to F.

3. In all other cases the Delayer defers to the Prover.

For the analysis we need two real parameters: $q \in (0,1)$ and $s \in (0, 3-3q)$. Since we need $\frac{1}{a} + \frac{1}{b} = 1$ we set $b = \frac{a}{a-1}$.

We now show that this strategy guarantees that the Delayer gets at least Dc points. Since the Delayer will *never* allow a monochromatic rectangle the game ends when there is some i, j such that

$$x_{ij1} = x_{ij2} = \cdots = x_{ijc} = F.$$

Who set these variables to F? Either at least qc were set to F by the Prover or at least $(1-q)c$ were set to F by the Delayer. This leads to several cases.

1. At least qc were set to F by the Prover. The Delayer gets at least $qc \lg a$ points.

2. At least $(1-q)c$ were set to F by the Delayer. For every k such that the Delayer set x_{ijk} to F there is an (i', j') (with $i \neq i'$ and $j \neq j'$) such that $x_{i'jk}$, $x_{ij'k}$, and $x_{i'j'k}$ were all set to T (we do not know by who). Consider the variables we know were set to T because Delayer set x_{ijk} to F. These variables all have the last subscript of k. Therefore these sets-of-three variables associated to each x_{ijk} are disjoint. Hence there are at least $3(1-q)c = (3-3q)c$ variables that were set to T. There are two cases. Recall that $s \in (0, 3-3q)$.

   (a) The Prover set at least sc of them to T. Then the Delayer gets at least $sc \lg(b) = sc \lg(a/(a-1))$ points.

   (b) The Delayer set at least $(s-(3-3q))c = (s+3q-3)c$ of them to T. If the Delayer is setting some variable $x_{i'j'k}$ to T it's because the Prover set rc others of the form $x_{i'j'k'}$ to F. These sets-of-rc-variables are all disjoint. Hence the Prover set at least $(s+3q-3)rc^2$ variables to F. Therefore the Delayer gets at least $(s+3q-3)rc^2 \lg a$ points.

We need to set $a \in (1, \infty)$, $q, r \in (0,1)$, and $s \in (0, 3-3q)$ to maximize the minimum of

1. $qc \lg a$

2. $sc \lg(a/(a-1))$

3. $(s+3q-3)rc^2 \lg a$

We optimize our choices by setting $qc \lg a = sc \lg(a/(a-1))$ (approximately) and thinking (correctly) that the c^2 term in $(s+3q-3)rc^2$ will force this term to be large when c is large. To achieve this we take

- $q = 0.56415$. Note that $3 - 3q = 1.30755$.
- $s = 1.30754$. Note that $s \in (0, 3-3q)$.
- $r = 0.9$. Note that $(s+3q-3)r = (0.00001) * 0.9 = 0.00009$. (Any value of $r \in (0,1)$ would have sufficed.)
- $a = 2.793200$
- $b = a/(a-1) = 1.557662$ (approximately)

Using these values we get $qc \lg a, sc \lg(a/(a-1)) \geq 0.836$. We want

$$(0.00009c^2) \geq 0.836c$$

$$(0.00009c) \geq 0.836$$

$$c \geq 9288$$

With this choice of parameters, for $c \geq 9288$, the Delayer gets at least $0.836c$ points. Hence any tree resolution proof of GRID(n, m, c) must have size at least $2^{0.836c}$. ∎

Note 5.7 The proof of Theorem 5.6 did not really use that the shape was a rectangle. We leave it to the reader to generalize the proof to other shapes. The resulting theorems will have other constants; however, the size will still have an exponential lower bound.

6 Lower Bounds on CP-Tree Res for GRID$(c+1, c\binom{c}{2}+1, c)$

By Lemma 4.3 the formula GRID$(c+1, c\binom{c}{2}+1, c) \notin$ SAT. Note that it's just barely not satisfiable since GRID$(c+1, c\binom{c}{2}, c) \in$ SAT. In this section we show that any Cutting Plane Tree Resolution proof that GRID$(c+1, c\binom{c}{2}+1, c) \notin$ SAT requires size $2^{\Omega(c^3/\log^2 c)}$.

Def 6.1 Let A be an integer valued matrix and $\vec{b}$ be an integer valued vector such that there is no 0-1 vector $\vec{x}$ with $A\vec{x} \leq \vec{b}$. We refer to this as $A\vec{x} \leq \vec{b} \notin$ SAT.

Any CNF-formula can be phrased in this form with only a linear blowup in size. For every variable x we have variables x and $\overline{x}$ and the inequalities

$$\begin{aligned} x + \overline{x} &\leq 1 \\ -x - \overline{x} &\leq -1 \end{aligned}$$

If C is a clause with literals $L_1, \ldots, L_k$ then we have the inequality

$$L_1 + \cdots + L_k \geq 1$$

In particular, the formulas GRID(n, m, c) can be put in this form.

6.1 Background on CP-Tree Resolution and Link to Communication Complexity

The definitions of Cutting Plane Proofs and Tree Cutting Plane Proofs are standard; however, we include them for completeness. The connection to communication complexity (Lemma 6.7) is by Impagliazzo et al. [5] (see also the book by Jukna [6] Lemmas 19.7 and 19.11).

Def 6.2 A *Cutting Planes Proof that* $A\vec{x} \leq \vec{b} \notin$ SAT (henceforth CP Proof) is a sequence of linear inequalities such that on each line you have either

1. One of the inequalities in $A\vec{x} \leq \vec{b}$ (called an AXIOM).
2. If $\vec{a}_i \cdot \vec{x} \leq c_i$ and $\vec{a}_j \cdot \vec{x} \leq c_j$ are on prior lines then $(\vec{a}_i + \vec{a}_j) \cdot \vec{x} \leq c_i + c_j$ can be on a line.

3. If $\vec{a} \cdot \vec{x} \leq c$ is on a prior line and $d \in \mathbb{N}$ then $d(\vec{a} \cdot \vec{x}) \leq dc$ can be on a line. (Also if $d \in \mathbb{Z} - \mathbb{N}$ then reverse the inequality.)

4. If $c(\vec{a} \cdot \vec{x}) \leq d$ is on a prior line then $\vec{a} \cdot \vec{x} \leq \left\lfloor \frac{d}{c} \right\rfloor$ can be on a line.

5. The last line is an arithmetically false statement (e.g., $1 \leq 0$).

6. For every Cutting Plane Proof there is a graph: (1) at the top there are the inequalities, $\vec{a}_i \cdots \vec{x} \leq c_i$, (2) when step 2 above is applied you have edges from inequalities, $\vec{a}_i \cdots \vec{x} \leq c_i$ and $\vec{a}_j \cdots \vec{x} \leq c_j$ to $(\vec{a}_i + \vec{a}_j) \cdot \vec{x} \leq c_1 + c_2$. (3) for the other steps there will be one edge in the obvious way.

It is easy to see that if there is a cutting planes proof that $A\vec{x} \leq \vec{b} \notin$ SAT then indeed $A\vec{x} \leq \vec{b} \notin$ SAT. The converse is also true though slightly harder to prove.

Def 6.3 The *size of a Cutting Plane Proof* is the number of nodes in the graph.

Def 6.4 A *Tree-like CP proof* is one whose graph is a tree.

We will connect the size of a Tree-like CP to communication complexity. Hence we need the following definition.

Def 6.5 1. For a communication problem $\mathcal{P}$, $D(\mathcal{P})$ denotes the deterministic communication complexity of $\mathcal{P}$ and $\mathrm{R}_\epsilon(\mathcal{P})$ denotes the randomized public coin communication complexity of $\mathcal{P}$ with error $\leq \epsilon$. Recall that such a protocol has both parties flip private coins and has probability of error $\leq \epsilon$.

Def 6.6 Let A be an integer valued matrix and $\vec{b}$ be an integer valued vector such that $A\vec{x} \leq \vec{b} \notin$ SAT. Let P_1, P_2 be a partition of the variables in $\vec{x}$. The Communication Complexity problem $\mathrm{FI}(A, \vec{b}, P_1, P_2)$ is as follows. (FI stands for *Find Inequality*.)

1. For every variable in P_1 Alice is given a value (0 or 1).

2. For every variable in P_2 Bob is given a value (0 or 1).

3. These assignments constitute an assignment to all of the variables which we denote $\vec{x}$.

4. Alice and Bob need to determine an inequality in $A\vec{x} \leq \vec{b}$ that is not true.

Lemma 6.7 *Let A be an integer valued matrix and $\vec{b}$ be an integer valued vector such that $A\vec{x} \leq \vec{b} \notin$ SAT. Let n be the number of variables in $\vec{x}$. If there is a partition P_1, P_2 of the variables such that, for all ϵ, $\mathrm{R}_\epsilon(\mathrm{FI}(A, \vec{b}, P_1, P_2)) = \Omega(t)$ then any tree-like CP proof of $A\vec{x} \leq \vec{b}$ requires size $2^{\Omega(t/\log^2 n)}$.*

6.2 Lemmas on Communication Complexity

Def 6.8

1. The *Hamming weight* of a binary string x, denoted $w(x)$, is the number of 1's in x.

2. The *Hamming distance* between two, equal-length, binary strings x and y, denoted $d(x, y)$, is the number of positions in which they differ.

Def 6.9 We define several communication complexity problems.

1. Let $\Sigma = \{1, \ldots, 2n-1\}$. PHPstr$_n$: Alice gets a string $x \in \Sigma^n$, and Bob gets a string $y \in \Sigma^n$. They are promised that, for all $i \neq j$, the letters x_i and x_j (resp. y_i and y_j) are distinct. By the PHP, there must exist at least one $(i, j) \in [n] \times [n]$ such that $x_i = y_j$. They are further promised that (i, j) is *unique*. *The goal is to find* (i, j). (Alice learns i, and Bob learns j.)

2. Let $\Sigma = \{1, \ldots, 2n-1\}$. PHPset$_n$: Alice gets a set $x \in \binom{\Sigma}{n}$, and Bob gets a set $y \in \binom{\Sigma}{n}$. By the PHP, there must exist at least one $\sigma \in \Sigma$ such that $\sigma \in x \cap y$. They are further promised that σ is *unique*. *The goal is to find* σ. (Both learn σ.)

3. PrMeet$_n$: Alice gets a string $x \in \{0,1\}^n$, and Bob gets a string $y \in \{0,1\}^n$ with $n = 2m-1$. They are promised that (1) $w(x) = w(y) = m$, (2) there is a *unique* $i \in [n]$ such that $x_i = y_i = 1$, and (3) for all $j \neq i$, $(x_j, y_j) \in \{(0,1), (1,0)\}$. *The goal is to find* i. (Both learn i.)

4. UM_n: (called the *universal monotone relation*) Alice is given $x \in \{0,1\}^n$, and Bob is given $y \in \{0,1\}^n$. They are promised that there exists i such that $x_i = 1$ and $y_i = 0$. *The goal is to find some such i.* (Both learn i.)

5. PrUM_n: This is a restriction of UM_n. They are additionally promised (1) $n = 2m - 1$ is odd, (2) $w(x) = m$, (3) $w(y) = m - 1$, and (4) $d(x, y) = 1$. Hence (a) there is a *unique* index $i \in [n]$ such that $x_i = 1$ and $y_i = 0$, (b) for all $j \neq i$, $(x_j, y_j) \in \{(0,0), (1,1)\}$, and moreover (c) these $(0,0)$'s and $(1,1)$'s occur in an equal number. *The goal is to find i.* (Both learn i.)

6. DISJ_n: Alice gets a string $x \in \{0,1\}^n$, and Bob gets a string $y \in \{0,1\}^n$. They need to *decide* if x and y intersect ($\exists i$ where $x_i = y_i$).

7. PrDISJ_n: $n = 2m + 1$ is odd. Alice gets a string $x \in \{0,1\}^n$, and Bob gets a string $y \in \{0,1\}^n$. They are promised that $w(x) = w(y) = m+1$ and $|x \cap y| \leq 1$. They need to *decide* if x and y intersect ($\exists i$ where $x_i = y_i$).

We will need the following notion of reduction.

Def 6.10 Let f, g be a communication problem. It can be a decision, a function, and/or a promise problem.

1. $f \leq_{\text{cc}} g$ if there exists a protocol for f that has the following properties.
   (a) The protocol may invoke a protocol for g once on an input of length $O(n)$.
   (b) Before and after the invocation, the players may communicate polylog bits.

The following lemma is obvious.

Lemma 6.11 *If $f \leq_{\text{cc}} g$ and $(\forall \epsilon)[\text{R}_\epsilon(f) = \Omega(n)]$ then $(\forall \epsilon)[\text{R}_\epsilon(g) = \Omega(n)]$.*

Lemma 6.12 *For all ϵ, $\text{R}_\epsilon(\text{PrUM}_n) = \Omega(n)$.*

Proof: Kushilevitz and Nisan [8] (Page 76) present a proof that $\text{DISJ}_n \leq_{\text{cc}} \text{UM}_n$. A closer examination of the proof shows that it also shows $\text{PrDISJ}_n \leq_{\text{cc}} \text{PrUM}_n$.

Kalyanasundaram and Schnitger [7] showed that, for all ϵ, $\text{R}_\epsilon(\text{DISJ}_n) = \Omega(n)$. Razborov [11] has a simpler proof where he only looks at inputs that satisfy the promise of PrDISJ_n. Hence he showed that, for all ϵ, $\text{R}_\epsilon(\text{PrDISJ}_n) = \Omega(n)$. From $\text{PrDISJ}_n \leq_{\text{cc}} \text{PrUM}_n$, $\text{R}_\epsilon(\text{PrDISJ}_n) = \Omega(n)$, and Lemma 6.11 the result follows. ∎

Lemma 6.13

1. $$\text{PrUM}_n \leq_{\text{cc}} \text{PrMeet}_n \leq_{\text{cc}} \text{PHPset}_{(n+1)/2}.$$
   (The last reduction only holds when n is odd.)

2. $$\text{PHPset}_n \leq_{\text{cc}} \text{PHPstr}_n.$$

3. *For all* ϵ $\text{R}_\epsilon(\text{PHPstr}_n) = \Omega(n)$. *(This follows from parts 1,2 and Lemmas 6.11, 6.12.)*

Proof:

$\underline{\text{PrUM}_n \leq_{\text{cc}} \text{PrMeet}_n}$: Map (x, y) to $(x, \overline{y})$. It's an easy exercise to see that this works.

$\underline{\text{PrMeet}_n \leq_{\text{cc}} \text{PHPset}_{(n+1)/2}}$: If $x \in \{0,1\}^n$ then $\text{SET}(x)$ is the subset of $\{1, \ldots, n\}$ that x represents as a bit vector. Map (x, y) to $(\text{SET}(x), \text{SET}(y)$. It's an easy exercise to see that this works. Example which we will continue in the next part: $(11001, 10110)$ maps to $(\{1,2,5\}, \{1,3,4\})$.

$\underline{\text{PHPset}_n \leq_{\text{cc}} \text{PHPstr}_n}$: $\Sigma = \{1, \ldots, 2n-1\}$. Alice gets $x \in \binom{\Sigma}{n}$, Bob gets $y \in \binom{\Sigma}{n}$. The sets x, y satisfy the promise of PHPset_n.

Alice (Bob) forms the string $x' \in \Sigma^n$ ($y' \in \Sigma^n$) which is the elements of x (y) written in order. It's an easy exercise to see that this works. Example: $\{1,2,5\}, \{1,3,4\}$ maps to $(125, 134)$.

∎

6.3 Lower Bound on CP-Tree Resolution for GRID$(c+1, c\binom{c}{2}+1, c)$

Theorem 6.14 *Let $A\vec{x} \leq \vec{b}$ be the translation of* GRID$(c+1, c\binom{c}{2}+1, c)$ *into an integer program. Any Tree-CP proof that $A\vec{x} \leq \vec{b} \notin$ SAT requires $2^{\Omega(c^3/\log^2 c)}$ size.*

Proof: We do the case where $c\binom{c}{2}+1$ is even (so $c \equiv 3 \pmod 4$). The other cases are similar but require slight variants of Lemma 6.13.

The variables in the equations $A\vec{x} \leq \vec{b}$ are of the form x_{ijk} where $1 \leq i \leq c+1$, $1 \leq j \leq c\binom{c}{2}+1$ and $1 \leq k \leq c$.

To partition the variables we first partition the set of ordered pairs of the first two coordinates.

Partition $[c+1] \times [c\binom{c}{2}+1]$ as follows:

$$Q_1 = \left\{(i,j) : (1 \leq i \leq c+1) \wedge \left(1 \leq j \leq \frac{1}{2}\left(c\binom{c}{2}+1\right)\right)\right\}$$

$$Q_2 = \left\{(i,j) : (1 \leq i \leq c+1) \wedge \left(\frac{1}{2}\left(c\binom{c}{2}+1\right)+1 \leq j \leq c\binom{c}{2}+1\right)\right\}$$

Partition the variables as follows.

$$P_1 = \{x_{ijk} : (i,j) \in Q_1\}$$

$$P_2 = \{x_{ijk} : (i,j) \in Q_2\}$$

We show that $\text{PHPstr}_n \leq_{\text{cc}} \text{FI}(A, \vec{b}, P_1, P_2)$ where $n = \Theta(c^3)$. Apply Lemma 6.13 to obtain that, for all ϵ, $\text{R}_\epsilon(\text{FI}(A, \vec{b}, P_1, P_2)) = \Omega(c^3)$. Note that the number of variables in $A\vec{x} \leq \vec{b}$ is $\Theta(c^4)$. Hence we can apply Lemma 6.7 to obtain that any Tree-CP proof that $A\vec{x} \leq \vec{b} \notin$ SAT requires $2^{\Omega(c^3/\log^2 c)}$ size.

We actually show that a restricted version of $\text{FI}(A, \vec{b}, P_1, P_2)$ *is* the PHPstr problem.

We restrict $\text{FI}(A, \vec{b}, P_1, P_2)$ to the case where every column has $c-1$ colors occurring once and the remaining color occurring twice. Hence one can view a coloring as a string of length $2m = c\binom{c}{2}+1$ over an alphabet of size $n = c\binom{c}{2}$.

Note that Alice and Bob each get a string of length m over an alphabet of size $n = 2m - 1$.

In order to find which inequality is violated Alice and Bob need to find which column they agree on (e.g., Alice's column i is the same as Bob's column j). This is precisely problem PHPstr problem. ∎

Lower bounds on Tree-CP proofs yield lower bounds on Tree-Resolution (with a constant factor loss) (see Prop 19.4 of the book by Jukna [6]). Hence we have the following.

Corollary 6.15 *Any Tree-resolution proof of* $\text{GRID}(c+1, c\binom{c}{2}+1, c) \notin \text{SAT}$ *requires* $2^{\Omega(c^3/\log^2 c)}$ *size.*

7 Open Problems

1. Let T be a theorem in this paper that mentions rectangles. What happens when *rectangle* is replaced with *square* in T?

2. Improve our FPT algorithm.

3. Prove that grid coloring extension problems starting with the empty grid is NP-complete. This may be the wrong question. We may need a new formalism for hardness.

4. Obtain exponential lower bounds for the size of resolution and cutting plane proofs of $\text{GRID}(n, m, c)$.

8 Acknowledgments

We thank Amy Apon, Doug Chen, Stasys Junka, Jon Katz, Clyde Kruskal, Nathan Hayes, and Rishab Pallepati for proofreading and discussion.

We thank Stasys Jukna whose marvelous exposition of the Prover-Delayer games and the tree-CP proofs inspired the second and third parts of this paper.

We thank Wing Ning Li for pointing out that the case of n, m binary, while it seems to not be in NP, is actually unknown.

We thank Daniel Marx for pointing out an improvement in the fixed parameter algorithm which we subsequently used.

We thank Tucker Bane, Richard Chang, Peter Fontana, David Harris, Jared Marx-Kuo, Jessica Shi, and Marius Zimand, for listening to Bill present these results and hence clarifying them.

References

[1] O. Beyersdorrff, N. Galesi, and M. Lauria. A lower bound for the pigeonhole principle in the tree-like resolution asymmetric prover-delayer games. *Information Processing Letters*, 110, 2010. `http://www.cs.umd.edu/~gasarch/resolution/resolution.html`.

[2] S. Fenner, W. Gasarch, C. Glover, and S. Purewal. Rectangle free colorings of grids, 2012. `http://arxiv.org/abs/1005.3750`.

[3] W. Gasarch. The 17×17 challenge. Worth $289.00. This is not a joke, 2009. November 30, 2009 entry on ComplexityBlog (Google Fortnow Blog).

[4] W. Gasarch. The 17×17 SOLVED! also 18×18, 2012. Feb 2, 2012 entry on ComplexityBlog (Google Fortnow Blog).

[5] R. Impagliazzo, T. Pitassi, and A. Urquhart. Upper and lower bounds for tree-like cutting planes proofs. In *Proceedings of the Ninth Annual IEEE Symposium on Logic in Computer Science,* Paris, France, 1994. `http://www.cs.toronto.edu/~toni`.

[6] S. Jukna. *Boolean function complexity: advances and frontiers.* Algorithms and Combinatorics Vol 27. Springer, New York, Heidelberg, Berlin, 2012.

[7] B. Kalyanasundaram and G. Schintger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5:545–557, 1992. `https://epubs.siam.org/doi/pdf/10.1137/0405044`.

[8] E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, Cambridge, England, 1997.

[9] S. Mahaney. Sparse complete sets for NP: Solution to a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25:130–143, 1982.

[10] P. Pudlak and R. Impagliazzo. A lower bound for DLL algorithms for SAT. In *Eleventh Symposium on Discrete Algorithms: Proceedings of SODA '00*, 2000.
`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.7363`.

[11] A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106:385–390, 1992.
`https://www.sciencedirect.com/science/article/pii/030439759290260M`.

[12] B. Steinbach and C. Posthoff. Extremely complex 4-colored rectangle-free grids: Solution of an open multiple-valued problem. In *Proceedings of the Forty-Second IEEE International Symposia on Multiple-Valued Logic*, 2012.

[13] B. Steinbach and C. Posthoff. The solution of ultra large grid problems. In *21st International Workshop on Post-Binary USLI Systems*, 2012.

[14] B. Steinbach and C. Posthoff. Utilization of permutation classes for solving extremely complex 4-colorable rectangle-free grids. In *Proceedings of the IEEE 2012 international conference on systems and informatics*, 2012.