DLINE JOURNALS

Signals and Telecommunication Journal

Print ISSN: 2278 – 6449
Online ISSN: 2278 – 6457

STJ 2025: 14 (1)
https://doi.org/10.6025/stj/2025/14/1/17-34

Novel Encoding Model for Asymptotically Greater Compactness

Bernardo Subercaseaux
Carnegie Mellon University
Pittsburgh, PA, USA

Marijn J.H. Heule
Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT

A packing k-coloring for a graph G = (V, E) is defined as a function that assigns colors from the set {1, ..., k} to the vertices in V. This assignment must ensure that any two vertices u and v that share the same color c are separated by a distance greater than c within the graph G. One of the key challenges in the area of packing colorings is to ascertain the packing chromatic number of the infinite square grid. Prior research has established that this number lies between 13 and 15. Our study enhances the lower limit to 14. Additionally, we introduce a novel encoding method that offers asymptotically greater compactness compared to those previously employed.

Keywords: Packing Coloring, SAT Solvers, EnCodings

Received: 4 October 2024, Revised 21 December 2024, Accepted 30 December 2024

Copyright: with Authors

1. Introduction

Automated reasoning techniques have been successfully applied to a wide variety of coloring problems, ranging from the classical computer assisted proof of the *Four Color Theorem* [1] establishing that 4 colors are enough to color any planar graph, to partial progress on the *Hadwiger-Nelson problem* [16] of computing the chromatic number of the graph with vertex set $\mathbb{R}^2$ and edges between points at euclidean distance exactly 1, and computing Ramsey-like numbers [12] that characterize the minimum size required for a graph to guarantee that all its colorings will contain certain local structures. In this context, this article's main focus is the use of automated reasoning techniques for improving the best known bounds on the *packing* chromatic number of the infinite

square grid. The notion of *packing coloring* was introduced in the seminal work of Goddard et al. [8][1], and has been extensively studied since [3]. Let us now jump to its definition.

1. Definition *Given a graph $G = (V, E)$, a packing k-coloring is a mapping* : $\varphi : V \to \{1, \ldots, k\}$ *such that for any pair of distinct nodes* $u, v \in V$ *and any color* $c \in \{1, \ldots, k\}$ *it holds that* $\varphi(u) = \varphi(v) = c \Rightarrow \text{dist}(u, v) > c$.

Year	Citation	Lower bound	Upper bound
2002	Goddard et al. [8]	9	23
2002	Schwenk [13]	9	22
2009	Fiala et al. [7]	10	23
2010	Soukal and Holub [17]	10	17
2010	Ekstein et al. [5]	12	17
2015	Martin et al. [10]	13	16
2017	Martin et al. [11]	13	15
2022	Our work	14	15

Table 1. Historical summary of the bounds known for $\chi_p(\mathbb{Z}^2)$

Note that the standard notion of coloring can be stated as requiring $\varphi(u) = \varphi(v) = c \Rightarrow \text{dist}(u, v) > 1$, which shows that packing colorings are a natural generalization. Moreover, the notion of chromatic number can be analogously defined for packing colorings as follows.

2. Definition *Given a graph $G = (V,E)$, define its packing chromatic number $\chi_p(G)$ as the minimum value of k such that G admits a packing k-coloring.*

For any $k \geq 4$, the problem of determining whether a graph G admits a packing 4-coloring is known to be NP-hard [3].

Example 3. Consider the infinite graph with vertex set $\mathbb{Z}$ and with edges between consecutive integers, which we denote as $\mathbb{Z}^1$. A packing 3-coloring is illustrated in Figure 1. On the other hand, by simple examination one can observe that it is impossible to obtain a packing 2-coloring for $\mathbb{Z}^1$.

[1] It was originally presented under the name of *broadcast coloring*, motivated by the problem of choosing broadcast frequencies for radio stations in a non-conflicting way [8], but the literature has preferred the name *packing coloring* since [3].

Figure 1. Illustration for a packing 3-coloring for $\mathbb{Z}^1$

Example 3 shows that $\chi_p(\mathbb{Z}^1) = 3$. Interestingly, the question of computing $\chi_p(\mathbb{Z}^2)$, where $\mathbb{Z}^2$ is the graph with vertex set $\mathbb{Z} \times \mathbb{Z}$ and edges between orthogonally adjacent points, has been open since the introduction of packing colorings by Goddard et al. [8]. The first bounds obtained were $9 \leq \chi_p(\mathbb{Z}^2) \leq 23$ [8], and before this work, the best bounds known where $13 \leq \chi_p(\mathbb{Z}^2) \leq 15$ by Martin et al. [11]. A summary of the progress until present is illustrated in Table 1. In what follows, we detail our approach to obtain a lower bound of 14, and present a theoretically compact encoding as well. For a survey in packing colorings, the reader can refer to that of Brešar et al. [3].

2. Direct Encoding and Basic Symmetry Breaking

Besides the lower bound of 10 proved by Fiala et al. [7], all other lower bounds have been proved with the help of computers. Among these, only that of Martin et al. [11] (i.e., the current best bound) was proved with the aid of SAT solvers. Our work continues along this line.

Proving lower bounds for the packing chromatic number of an infinite graph usually relies on the following trivial proposition.

Proposition 4 ([8]). *Let G be a graph, and H be a sub-graph of G. Then $\chi_p(H) \leq \chi_p(G)$.*

For example, Martin et al. conclude that $\chi_p(G) \geq 13$ by proving that a certain graph $H \subset \mathbb{Z}^2$ (which consists on a 14×14 grid, with the number 9 forced in position $(7, 12)$) does not admit a packing 12-coloring, thus implying that $\chi_p(\mathbb{Z}^2) \geq \chi_p(H) > 12$.

The *direct* encoding for determining whether a finite graph $H = (V, E)$ admits a packing k-coloring for some k is as follows.

1. Create variables $x_{v,i}$ for each $v \in V$ and $1 \leq i \leq k$, stating that vertex v receives color i.

2. Create a clause $x_{v,1} \vee x_{v,2} \vee \ldots \vee x_{v,k-1} \vee x_{v,k}$ for each vertex $v \in V$, implying that every vertex will be assigned at least some color.

3. For each pair of vertices u, v and color $i \in \{\text{dist}(u, v), \ldots, k\}$, create a clause $\bar{x}_{u,i} \vee \bar{x}_{v,i}$ for bidding that both u and v get color i.

Let us now denote by $B(v, n) := \{u \in V(\mathbb{Z}^2) \mid \text{dist}(u, v) \leq n\}$ the n-radius ball in $\mathbb{Z}^2$, and recall that $|B(v, n)| = 2n^2 + 2n + 1 = O(n^2)$ for any v. Therefore, to encode whether a given ball $B_n := B((0, 0), n)$ admits a packing k-coloring, the direct encoding requires $O(n^2k)$ variables, and its number of clauses is

$$O(n^2) + \sum_{i=1}^{k} \left| \{(u, v) \in V(B_n)^2 \mid 0 < \text{dist}(u, v) \leq i\} \right|$$

$$= O(n^2) + \sum_{i=1}^{k} \sum_{u \in V(B_n)} |\{v \in V(B_n) \setminus \{u\} \mid \text{dist}(u,v) \leq i\}| / 2$$

$$= O(n^2) + \sum_{i=1}^{k} n \cdot O(i^2) = O(n^2 + nk^3).$$

Moreover, let $B_{n,k}$ represent whether B_n admits a packing k-coloring. Then, let $D_{n,k}$ be the SAT instance created with the direct encoding for $B_{n,k}$ and let $D^+_{n,k}$ be equal to $D_{n,k}$ but forcing its central vertex to be assigned color $\min(n, k)$, choice that we justify later on. Namely, adding the unit clause $x_{(0,\,0),\,\min(n,k)}$. Figure 2 illustrates a satisfying assignment for $D+$ 3, 7.

Proposition 5 *If* $\chi_p(\mathbb{Z}^2) \geq \min(n, k)$ *for some pair* (n, k)*, and* $D+_{n,k}$ *is unsatisfiable, then* $\chi_p(\mathbb{Z}^2) \geq k + 1$.

Proof If $k \leq n$ argue as follows. Assume, expecting a contradiction, that $\mathbb{Z}^2$ admits a packing k-coloring φ. As $\chi_p(\mathbb{Z}^2) \geq k$, every φ must assign at least one vertex $v \in V(\mathbb{Z}^2)$ to color k (otherwise φ would be a packing $(k - 1)$-coloring). Now, note that no other vertex besides v can receive color k in $B(v, n)$, and thus φ restricted to $B(v, k)$ provides a satisfiable assignment for $D^+_{n,k}$ which contradicts the hypothesis. On the other hand, if $k > n$, assume a packing k-coloring φ and as $\chi_p(\mathbb{Z}^2) \geq n$, there must exist a vertex $v \in \mathbb{Z}^2$ such that $\varphi(v) \geq n$, Let v be such that $\varphi(v) \geq n$ is minimized. Then, consider the ball $B(v, n)$ and note that either $\varphi(v) = n$ or otherwise $\varphi(v) > n$ and thus $\{u \in B(v, n) \mid \varphi(u) = n\} = \phi$, by minimality of $\varphi(v)$. In either case, φ restricted to $B(v, n)$, but assigning color n to v, provides a satisfiable assignment for $D^+_{n,k}$.

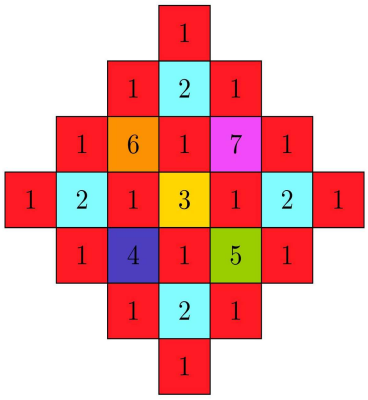


Figure 2. Illustration of a satisfying assignment for $D+_{3,7}$

As we trivially have that if $D_{n,k}$ is unsatisfiable then $\chi_p(\mathbb{Z}^2) \geq k$, one can prove for example that $\chi_p(\mathbb{Z}^2) \geq 12$ by showing that both $D_{4,7}$ and $D^+_{7,11}$ are unsatisfiable instances. While this endeavor took Ekstein et al. [5] 120 days of brute force computation, admittedly with hardware from 2009, a current personal computer is able to prove the unsatisfiability of $D_{4,7}$ and $D^+_{7,1}$ in less than an hour by using some further optimizations we will present later. Intuitively, the reason to force $\min(n, k)$ in the center is that it is the smallest color we can assign that guarantees no more occurrences besides the center. As in packing colorings smaller colors have *densities* (i.e., fractions of the vertices receiving said colors) that are greater or equal than those of higher colors [3],

forcing a color c to appear only once is more useful the smaller c is, as we are reducing from a higher original density. Note as well that, as opposed to the previous work that used rectangles to provide lower-bounds [5, 11], our basic shape is an l_1-ball. Let us present an intuitive reason for why this is a good idea. Consider the following property: a subgraph $H \subset \mathbb{Z}^2$ is said to be (n, k)-single, for $n \leq k$ if there is a vertex $v \in V(H)$ such that assigning color n to v guarantees that in no packing k-coloring of H other vertex $u \neq v$ can be assigned color n. Now note that, for any pair (n, k), it happens that Bn is the smallest (n, k)-single graph in terms of its number of vertices.

In order to quantitatively understand effect of using l_1-balls instead of squares, as well as the difference between $D+_{n,k}$ and $D_{n,k,}$ we study the time required to prove a given lower bound $\chi_p(\mathbb{Z}^2) \geq k$ with each of them, under the same hardware. We will use notation S_n to refer to the square grid of $n \times n$, and $S_{n,k}$ to the direct encoding for S_n with k colors. Moreover, to understand the impact of forcing, let us define $S+_{n,k}$ to be equal to $S_{n,k,}$ but with an added unit clause enforcing that the value at the center of the grid, namely at position $(\lfloor n/2 \rfloor, \lfloor n/2 \rfloor)$ to contain value $\min(k, \lfloor n/2 \rfloor)$.

Table 2 presents results comparing the runtime of the different alternatives. It can be appreciated that although there is not a significant difference between shapes alone, the forcing creates a substantial difference. This can be interpreted as a rather trivial form of symmetry breaking. Empirically we found that l_1-balls appeared to work better with the further optimizations we introduce, and thus we stick with them.

2.1 Experimental Setup

In terms of software, experiments from Table 2 were ran on state-of-the-art solver CaDiCaL [2], while experiments with cube-and-conquer, as those in Table 3 and Table 4 were ran using iLingeling because it supports incremental

lower bound	Requires UNSAT of (# vertices)				Time			
	Square	Square + force	l_1-ball	l_1-ball+ force	Square	Square+ force	l_1-ball	l_1-ball+ force
5	$S_{4,4}$ (16)	$S+_{4.4}$ (16)	$D_{2,4}$ (13)	$D+_{1,4}$ (5)	0.00s	0.00s	0.00s	0.00s
6	$S_{5,5}$ (25)	$S+_{5,5}$ (25)	$D_{3,5}$ (25)	$D+_{2,5}$ (13)	0.00s	0.00s	0.01s	0.00s
7	$S_{5,6}$ (25)	$S+_{5,6}$ (25)	$D_{4,6}$ (41)	$D+_{3,6}$ (25)	0.04s	0.04s	0.04s	0.01s
8	$S_{6,7}$ (36)	$S+_{6,7}$ (36)	$D_{4,7}$ (41)	$D+_{4,7}$ (41)	0.22s	0.04s	0.20s	0.06s
9	$S_{7,8}$ (49)	$S+_{7,8}$ (49)	$D_{5,8}$ (61)	$D+_{4,8}$ (41)	7.00s	0.28s	7.56s	0.28s
10	$S_{9,9}$ (81)	$S+_{8,9}$ (64)	$D_{6,9}$ (85)	$D+_{5,9}$ (61)	147.52s	11.82s	159.21s	12.22s
11	$S_{10,10}$ (100)	$S+_{9,10}$ (81)	$D_{7,10}$ (113)	$D+_{5,10}$ (61)	2.56hrs	324.49s	3.03hrs	229.76s
12	$S_{12,11}$ (144)	$S+_{11,11}$ (121)	$D_{8,11}$ (145)	$D+_{,11}$ (85)	>24hrs	4.92hrs	>24hrs	5.76hrs

Table 2. Comparison of basic approaches for showing lower bounds on $\chi_p(\mathbb{Z}^2)$using CaDiCaL

solving [9]. In terms of hardware, all our experiments were run in the Bridges2 cluster of the Pittsburgh Supercomputing Center [4], which has the following specifications:

Two AMD EPYC 7742 CPUS, each with

- 64 cores
- 2.25-3.40GHz
- 256MB L3
- 8 memory channels
- 512GB of RAM
- NVMe SSD (3.84TB)
- Mellanox ConnectX-6 HDR Infiniband 200Gb/s Adapter.

3. Proving that 14 is a Lower Bound

Given that runtime increases exponentially with respect to the number of colors (see Table 2), to solve instance $D+_{12,13}$ within a reasonable amount of time, say 48hrs of computation (including parallelism), we required some further optimizations.

In particular, we followed the *cube-and-conquer* approach [9]. In a nutshell, cube-andconquer is based on constructing a tautological DNF formula $\varnothing = C_1 \vee \ldots \vee C_m$, with cubes $C_1, \ldots, C_m$, and then using the following identity for any formula ψ:

$$\mathrm{SAT}(\psi) \iff \mathrm{SAT}(\psi \wedge \phi) \iff \mathrm{SAT}\left(\bigvee_{i=1}^{m} (\psi \wedge C_i)\right).$$

This equivalence means that we can solve instance ψ by solving a certain numbers of instances of the form $\psi \wedge C_i$ These instances, if the cubes C_i are properly designed, should be much easier to solve than the original instance ψ. Moreover, it is clear from its definition that this approach is well-suited for parallel computation.

Let us now describe the construction of cubes that we used, which is also presented as pseudocode in Algorithm 1. Let F and d be fixed integers. Then, our cubes will be based on forcing up to F colors in the ball $B((0, 0), d)$. More precisely, consider an instance $D+_{n,k,}$ and let $c = \min(n, k)$ be the color forced at the center. Then, let $K = \{k, k-1, \ldots, k-F\} \setminus \{c\}$ the set of the F highest colors without considering the one forced at the center. Now, for every value $f \in \{F, \ldots, 0\}$, we will create cubes forcing f colors in the following way. For every ordered set $O = (o_1, \ldots, o_f)$ consisting of f vertices in $B((0, 0), d) \setminus \{(0, 0)\}$, and every permutation $k_1, \ldots, k_f$ of every subset $K' \subseteq K$, $|K2| = f$, create a cube that forces each vertex o_i to receive color k_i, and that asserts as well that colors in $K \setminus K'$ are not assigned to any vertex in $B((0, 0), d)$.

Example 6 Figure 3 illustrates a cube created for solving the instance $D+_{5,9,}$ with parameters $F = 4$ and $D = 3$. In particular $K = \{6, 7, 8, 9\}$, and the illustrated cube corresponds to $f = 3$, $K' = \{6, 8, 9\}$ and $O = ((-1, 2), (1, 1), (0,-1))$.

Let us prove that this construction forms indeed a tautology, while we also point out that we checked with a SAT-solver that the generated cubes for the instance we ran form indeed a tautology.

Lemma 7 The construction of cubes presented in Algorithm 1 results in a tautology.

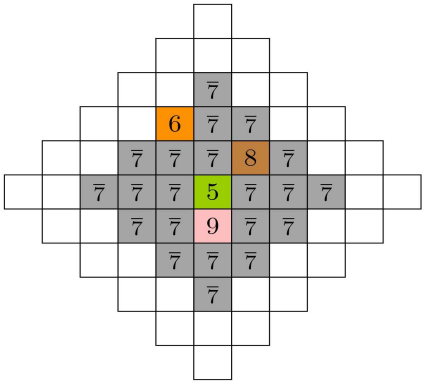


Figure 3. Illustration of a cube for instance D+ 5,9. The ball $B((0, 0), 3)$ is colored with light gray,and notation $\overline{7}$ is used to indicate that a given cell is forced to not contain color 7

```
1.1  K ← {k, k − 1, . . . , k − F} \ {min(n, k)}
1.2  cubes ← ∅
1.3  for f ∈ {F, F − 1, . . . , 0} do
1.4      for O ⊆ B((0, 0), d) \ {(0, 0)}, |O| = f do
1.5          for K′ ∈ Permutations(K, f) do
                 ▷ Assume Permutations(K, f) returns a list with all the different f-sized ordered sets
                   of K.
1.6              cube ← ∅
1.7              for i ∈ {1, . . . , f} do
1.8                  (a, b) ← o_i
1.9                  k ← K′_i
1.10                 cube ← cube ∪ {x_{(a,b),k}}
1.11             for t ∈ K \ K′ do
1.12                 for v ∈ B((0, 0), d) \ O do
1.13                     cube ← cube ∪ {¬x_{v,t}}
1.14             cubes ← cubes ∪ {cube}
```

Algorithm 1. CubeConstruction(*n, k, d, F*)

Proof Consider the cube constructed when f takes value 0. That cube states that no vertex $v \in B((0, 0), d)$ receives any color in K. If that cube holds, then the DNF is satisfied. Otherwise, some vertices in $B((0, 0), d)$ must receive colors in K. Let

$$K^* = \{k \in K \mid \exists v \in B((0,0), d), x_{v,k} = 1\},$$

and note that K^* is non empty because the cube associated to value $f = 0$ does not hold. Then, let O^*be an ordered list of vertices getting different colors in K^*. As $|K| = F$, we have $|K^*| \leq F$, and thus O^* contains at most F vertices. These implies that some cube is generated exactly with the values $f = |K^*|$, $K' = K^*$ and $O = O^*$, and is by definition satisfied, thus satisfying the DNF.

3.1 Symmetry Breaking

To further optimize performance, there is a simple kind of symmetry breaking we can apply.Indeed, note that the ℓ_1-ball presents a natural symmetry with respect to 4 different axis,which will allows to asymptotically reduce the number of cubes to consider by a factor of 8.

More precisely, it suffices to consider cubes in which the largest number forced appearsin a particular octant, namely $\{(i, j) \mid i \geq 0, j \geq 0, i \geq j\}$. Thus, we can simply optimize Algorithm 1 by considering only cubes whose largest forced color lies in said octant. This is illustrated in Figure 4. Despite its simplicity, this form of symmetry breaking had not been used in the past to the best of our knowledge.

3.2 Number of Generated Cubes

The number of cubes generated by Algorithm 1 is an important parameter of our approach; more cubes usually imply that each cube is easier to solve, up to a certain point at which the sheer number of cubes becomes the dominant factor in runtime and thus performance decreases. Moreover, this analysis can become even more complex in the presence of parallel computation.

First, let us analyze the number of cubes asymptotically. Directly from Algorithm 1 it follows that the number of cubes for a given value of f is exactly

$$\binom{|B((0,0),d) \setminus \{0,0\}|}{f} \binom{|K|}{f} f! = \binom{2d^2+2d}{f} \binom{F}{f} f!,$$

and thus the total number of cubes is

$$\sum_{f=0}^{F} \binom{2d^2+2d}{f} \binom{F}{f} f!.$$

To obtain a simpler expression we can use the standard bound $\binom{a}{b} \leq (ea/b)^b$, and note that $f = F$ is the largest term in the sum, which implies that previous formula is asymptotically bounded above by $F(e^2(2d^2 + 2d)F) \leq F(3ed)^{2F}$.

d	*F*	# cubes	# cubes w.SB	Time cubes w/o SB			Time cubes w/ SB		
				total	average	max	total	average	max
2	2	157	40	126.71s	0.80s	12.35s	36.53s	0.91s	10.09s
2	3	1753	439	145.86s	0.08s	2.75s	38.43s	0.08s	2.56s
2	4	18001	4501	183.23s	0.01s	0.48s	45.48s	0.01s	0.33s
3	2	601	126	213.10s	0.35s	1.32s	45.25s	0.33s	1.47s
3	3	13873	2891	400.37s	0.02s	0.39s	82.90s	0.02s	0.25s
3	4	307009	63961	1204.19s	0.00s	0.24s	199.41s	0.00s	0.05s

Table 3. Illustration of the effect of the different parameters in the number of cubes, and the derived runtime, for instance $D+_{5,10}$. Symmetry breaking is abbreviated as SB

d	F	# cubes w. SB		Time w. SB		
			Total	Wall clock	Avg. cube	Max cube
2	2	40	3280.3s	1995.7s	82.4s	1994.6s
2	3	439	2845.0s	607.0s	6.7s	599.6s
2	4	4501	3038.5s	197.2s	0.7s	180.1s
2	5	42256	3549.1s	47.5s	0.08s	20.8s
3	2	126	3141.0s	298.5s	26.6s	297.3s
3	3	2891	3772.1s	79.6s	1.3s	51.5s
3	4	63691	6332.4s	59.9s	0.1s	8.6s
3	5	1354726	13216.4s	122.3s	0.01s	2.6s

Table 4. Illustration of the effect of the different parameters in the number of cubes, and the derived runtime, for instance D+7,11. Symmetry breaking is abbreviated as SB. All instances were ran on a 128 cores machine

By applying the symmetry breaking procedure described in Section 3.1, this is further reduced by a constant factor that converges to 8 asymptotically. Table 3 presents the number of cubes generated under different parameters for instance $D^+_{5,10,}$ while Table 4 presents results for instance $D^+_{6,11}$. Note that the best sequential time for $D^+_{6,11}$ with our cube-and-conquer approach is 2845 seconds, which represents more than a 5x improvement with respect to the sequential execution shown in Table 2. Moreover, the speed-up we obtain from parallelism is almost linear (i.e., best possible), as 128 cores allow for the best parallel runtime to be a 60x improvement over the best sequential runtime (47s vs. 2845s).

Based on these experiences, we approached $D^+_{12,13}$ by setting $F = 5$ and $d = 3$. Our rationale for this is twofold. On the one hand, it can be appreciated from comparing Table 3 and Table 4 that, as the instance to solve gets larger, the value of $d + F$ for its optimal parameters increase. For example, for instance $D+_{6,11,}$ the best combinations are $d = 2$, $F = 5$ followed by $d = 3$, $F = 4$, so the optimal value of $d + F$ appears to be 7. More in general the runtime of in-parallel execution can be dominated by the time of the hardest cubes, and thus reducing said time usually improves wall clock time. Both the increase of F and d contribute towards this goal. On the other hand, it is clear in Table 4 that the total execution time gets larger, and at $d = 3$, $F = 6$ there are more than 100 million cubes, thus making the total execution time unmanagable.

3.3 Partioning the Last Cube

In order to tackle $D^+_{12,13}$ we introduced yet another optimization. One can see in practice that the cubes constructed by Algorithm 1 get progressively harder as f is decreasing. In particular, the cube in which $f = 0$, and thus the only condition the cube imposes is that no vertex in $B((0, 0), d)$ receives a color in K, is usually the one that takes the most time to solve. Especially in the context of parallel execution, runtime can be dominated

by the time it takes to solve the hardest cube, which motivates us to reduce its difficulty. For this purpose we introduced a last optimization that we refer to as *last cube partitioning* that is described next.

The cube constructed by Algorithm 1 with $f = 0$ is stating that there is a coloring in which no vertex in $B((0,0), d)$ receives a color in K. Such a coloring can be conditioned on whether it assigns the largest color outside of K to some vertex in $B((0,0), d)$ or not. If it does not, then we can condition on whether it assigns the second largest color outside of K to some vertex in in $B((0, 0), d)$, and so on. This way, the hardest cube is broken into $B((0,0), d)|$ cubes, of which $|B((0, 0), d) \setminus \{(0, 0)\}|$ consist of forcing the largest color outside of K to the different positions in $B((0, 0), d) \setminus \{(0,0)\}$, and the last one (which can be broken down in the same fashion), states that colors in $K \cup \{\max(Kc)\}$ are not assigned to any vertex in $B((0, 0), d) \setminus \{(0, 0)\}$. We empirically found that doing two steps of this recursive partitioning was enough to replace the last cube by a series of cubes that take less than a second each. Part of our future work includes further experimentation with this optimization.

3.4 Solving $D+_{12,13}$

Using all optimizations discussed so far, included symmetry breaking, we were able to solve instance $D+_{12,13}$ in less than 48 hours. As our symmetry breaking process has not been formally verified yet (which we leave for future work), we also solved every cube associated to $D^{+}_{12,13,}$ increasing total runtime by a factor of 5. As it was known before that $\chi_p(\mathbb{Z}^2) \geq 13$ [11], we use Proposition 5 together with the unsatisfiability result obtained for $D+_{12,13}$ to prove our main theorem.

Theorem 8 $\chi_p(\mathbb{Z}^2) \geq 14$

Let us now present some data regarding the execution over instance $D+_{12,13}$. Because of parallelization over 128 cores (see Section 2.1), we report both the total execution time (meaning the sum of the time every single cube took) and wall clock time. Considering symmetry breaking, there were a total of 1354741 many cubes. The total execution time was 3200hrs., while wall clock time was 45hrs. Moreover, the average time spent per cube was only 8.50s., while the hardest cube took 30hrs. Figure 5 shows how progress (in terms of the number of cubes solved) evolves over time. Figure 6 shows statistics on the time spent per cube.

4 A Recursive Encoding

This section presents a boolean encoding for $B_{n,k}$ that is much more asymptotically compactthan $D_{n,k,}$ and how to combine it with the direct encoding to obtain one that is either equally or more compact than the direct one in any case.

Theorem 9 *There is an encoding $C_{n,k}$ for Bn,k that uses $O(n^2k \lg k)$ variables and clauses.*

The encoding, which we shall call *recursive encoding*, is composed of two different kinds of variables:

$x_{v,t}$ representing that vertex v gets color t. ($v \in B_{n,}$ $t \in [k]$).

$f_{v,t,r}$ representing that no vertex in $B(v, 2^r)$ gets color t. ($v \in B_{n,}$ $t \in [k]$, $r \in [\lg_2(2k)]$).

Note immediately that the number of variables matches the promised bound. Just as the direct encoding, we require for every vertex v a clause stating that it will receive at least one color. The fundamental difference

with the direct encoding will be in the way that conflicts are handled. We want to enforce that $x_{v,t}$ implies no vertex in $B(v, t) \setminus \{v\}$ receives color t. For this purpose we will use a constant number of the $f_{v,t,r}$ variables, for an appropriate set of choices for v and r.

Let us start by enforcing that the $f_{v,t,r}$ variables achieved their desired semantic. We can do so by defining $f_{v,t,r}$ in terms of the x variables for $r = 1$, and then using the following recursive implication:

$$f(i, j), t, r \rightarrow f(i\, 2^{r-1}, j), t, r-1 \wedge f(i + 2^{r-1}, j), t, r\,1 \wedge\ f(i,\ j\text{-}2^{r-1}), t, r\text{-}1 \wedge f(i, j + 2^{r-1}), t, r\text{-}1$$

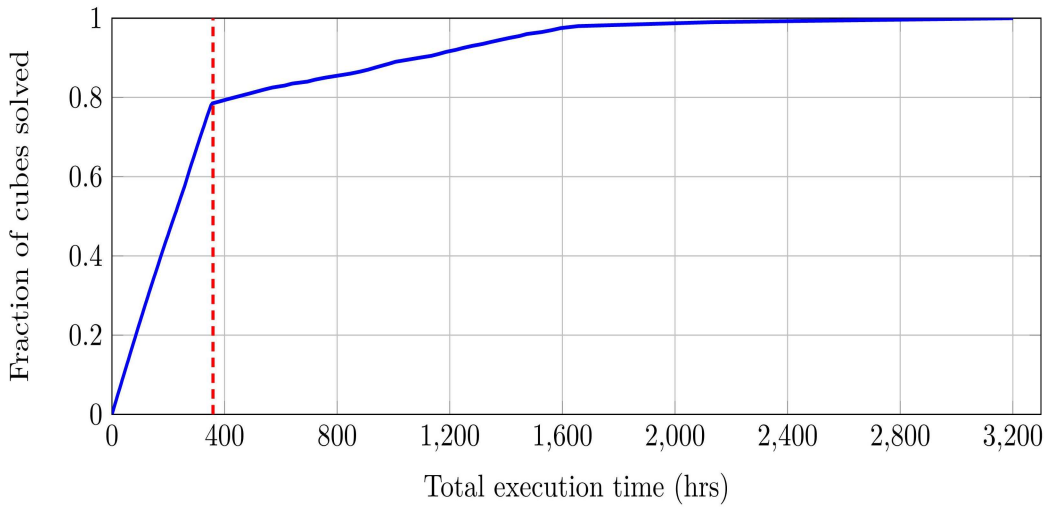


Figure 5. Depiction of the fraction of cubes solved over total execution time. A red dashed line at 359 hours shows the point at which almost all cubes with $f = 5$ have been solved

which is illustrated in Figure 7.

With the goal of simplifying the exposition, let us introduce variables $w_{v,t,r}$ that will not be part of the actual encoding, and thus can be understood as shorthands for an expression we will define later. Semantically, $w_{v,t,r}$ represents that no vertex in $B(v, r)$ gets color t.

Lemma 10 For a fixed vertex v and color t, it is possible to encode that no vertex in $B(v, t) \setminus \{v\}$ receives color t with using only $O(1)$ clauses, without counting the clauses defining the fv,t,r variables described above.

Proof First, if $t \leq 2$, using the direct-encoding for this satisfies the statement. We will thus assume $t \geq 3$. As the construction will depend on the parity of t, Figure 8 illustrates a decomposition for odd values of t, while Figure 9 illustrates a decomposition for even values
of t.

By using at most 12 variables $w_{v,t,r}$ we can enforce that no vertex in $B(v, t) \setminus \{v\}$ receives color t. In particular, let $v = (i, j) \in V(\mathbb{Z}^2)$. Then, enforce 4 constraints of the form $w_{(i+\Delta_i, j+\Delta_j), t, \lfloor \frac{t-1}{2} \rfloor}$ for $(\Delta_i, \Delta_j) \in \{(-1, 0), (1, 0), (0, -1), (0, 1)\}$ we call these primary constraints.

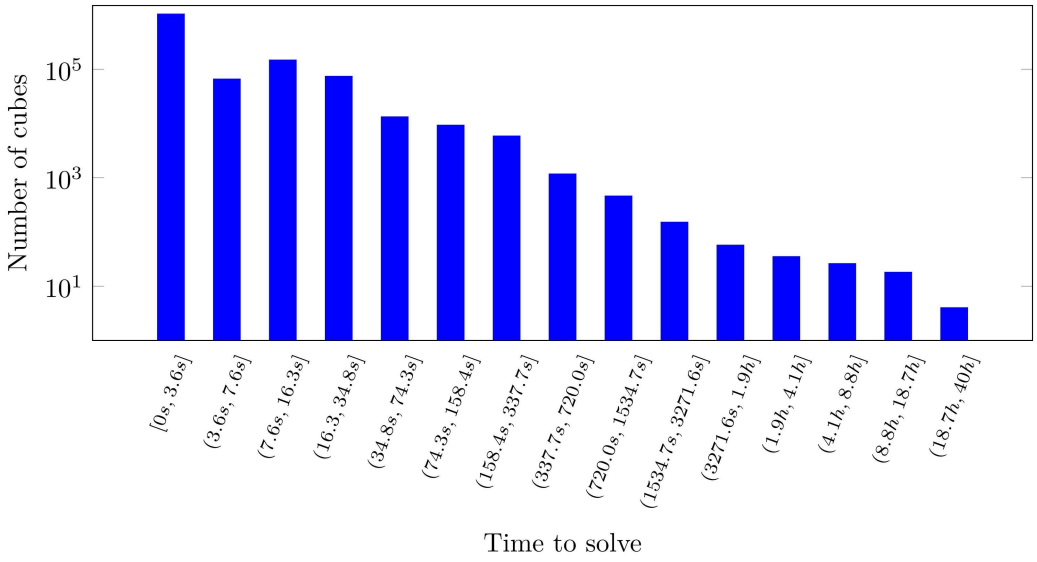


Figure 6. Categorization of the time spent per cube. The times are separated into 14 intervals in a geometric rogression from 0 seconds up to 40 hours, and for each interval the number of cubes whose solution time lies within the interval is displayed

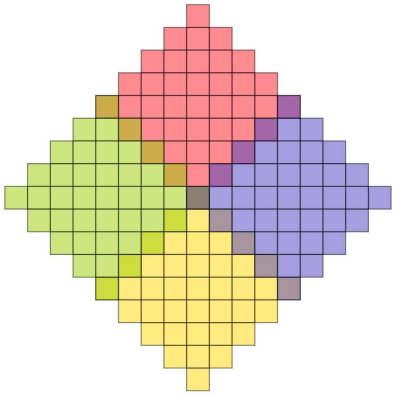


Figure 7. Illustration for $r = 3$ of the recursive decomposition for the f variables

Now, to fill in the gaps between the regions, we will enforce secondary constraints. Let us first define $t' = (\lfloor t/2 \rfloor + 1)/2$ and $t'' = \lfloor (t - 1)/2 \rfloor$, and now we proceed to detail the secondary constraints:

1. $w_{(i-\lfloor t' \rfloor, j-\lfloor t' \rfloor, t, t''}$

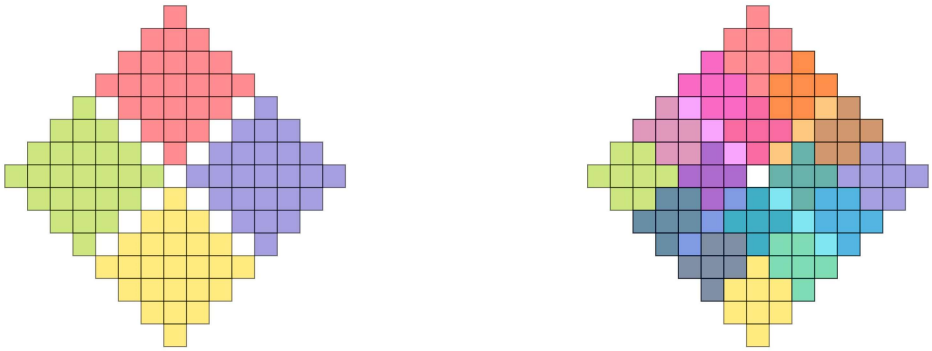
2. $w_{(i-\lfloor t' \rfloor, j-\lfloor t' \rfloor, t, t''}$

3. $w_{(i+\lfloor t' \rfloor, j-\lfloor t' \rfloor, t, t''}$

4. $w_{(i+\lfloor t' \rfloor, j-\lfloor t' \rfloor, t, t''}$

Assuming the $w_{v,t,r}$ constraints enforce the semantic of forbidding color t in $B(v, r)$,then the previous decomposition does indeed enforce exactly that no vertex in $B(v, t) \setminus \{v\}$ receives color t for odd values of t.

However, if t is even, as illustrated in Figure 9, we will need 4 new constraints, that we can call tertiary constraints, and simply enforce that the 4 vertices orthogonally adjacent to v do not receive color t.



(a) Illustration of the primary constraints

(b) Illustration of the primary and secondary constraints

Figure 8. Illustration of an odd decomposition for $t = 7$

It remains to get rid of the assumption about the $w_{v,t,r}$ variables. This assumption is in fact not too strong, a each $w_{v,t,r}$ constraint can be enforced through 4 clauses using the $f_{v,t,r}$ variables. If $r = 2^p$, then $w_{v,t,r}$ is semantic ally equal to $f_{v,t,p}$ and nothing needs to be done. Otherwise, let r^* be the largest power of 2 that is smaller than r. Observe that $r^* \geq r/2$. Then, if $v = (i, j)$, we can define $w_{v,t,r}$ as

$$w_{v,t,r} \leftrightarrow \bigwedge_{(\Delta_i,\ \Delta_j)\in\{(-1,\ 0),(1,\ 0),(0,\ 1),(0,\ -1)\}} f\left(i+\Delta_i(r-r^*), j+\Delta_j(r-r^*)\right), t, p^* .$$

As the whole-decomposition of the condition we wish to enforce uses 8 of the w constraints, each of which will be enforced through 4 clauses using the f variables, this requires a total of 32 clauses if t is odd, and 36 clauses if t is even.

We can see now that the total number of clauses matches the promised bound. Indeed, each $f_{v,t,r}$ only requires 4 clauses to be properly defined, incurring into $O(n^2k \lg k)$ clauses, each vertex v requires 1 big clause stating it gets at least a color, incurring into $O(n^2)$ clauses, and the result of Lemma 10 implies that each variable $x_{v,t}$ avoids conflicts in $B(v, t)$ through a constant number of clauses, incurring into $O(n^2k)$ clauses. Thus we conclude that the recursive encoding uses $O(n^2k \lg k)$ variables and clauses.

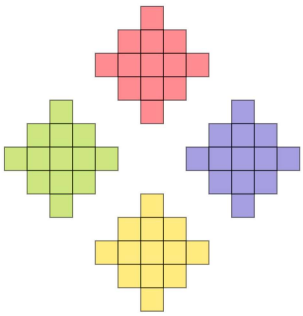
4.1 Compact Encoding

Although the recursive encoding presented above requires asymptotically fewer clauses, it has a larger constant factor than the direct encoding, and thus it only improves the size of the encoding from a given point onwards. That is, for every size n, there exists a value $p(n)$, such that it is more efficient to encode conflicts for colors above $p(n)$ recursively. If $p(n) > k$, then the direct encoding is best for a given $B_{n,k}$. This allows to define the compact

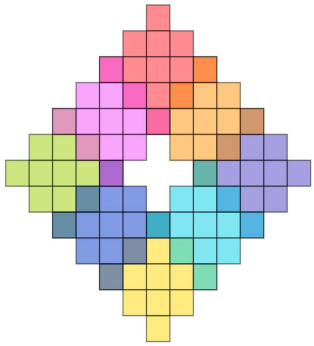
encoding as that in which conflicts are encoded recursively only for colors in which this improves with respect to the direct encoding. Guaranteeing that the number of clauses created by the compact encoding for $B_{n,k}$ is at most that of the direct encoding for $B_{n,k}$.

4.2 Comparing the Encodings

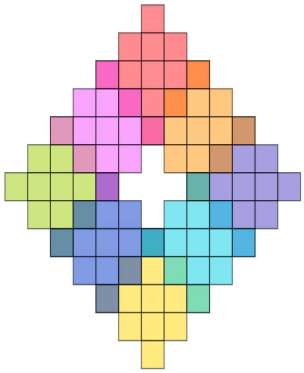
Table 5 compares the number of variables and clauses for different $r \times r$ grids with k colors.We chose to implement the recursive encoding over grids instead of l_1-balls for simplicity of implementation. It can be observed that the compact encoding improves the most in larger instances, such as when trying to prove a lowerbound of 13 or 14. In our experiments up to 13, the compact encoding does not provide a significant speed up in terms of runtime, and thus so far its interest is mostly theoretical.



(a) Illustration of the primary constraints



(b) Illustration of the primary and secondary constraints



(c) Illustration of the primary, secondary and tertiary constraints

Figure 9. Illustration of an even decomposition for $t = 6$

5. Directed Graphs for Handling Infinite Trees

Another family of infinite graphs for which the packing chromatic number has been studied is that of infinite perfect n-ary trees T_n [3]. It was shown by Sloper that $\chi_p(T_2) = 7$, and that $\chi_p(T_n) = \infty$ for $n \geq 3$. Moreover, Fiala and Golovach have shown that computing $\chi_p(T)$ is NP-hard for an arbitrary tree T [6]. We will show in this

(r, k)	Direct encoding		Recursive encoding		Compact encoding	
	# variables	# clauses	# variables	# clauses	# variables	# clauses
(7, 8)	392	10325	10072	29573	392	10325
(9, 9)	729	27009	17629	54450	729	27009
(10, 10)	1000	44848	25389	78960	1000	44848
(12, 11)	1584	90520	37521	123220	1584	90520
(14, 12)	2352	165088	53005	182544	24256	148508
(17, 13)	3757	327161	76081	277006	39037	245866
(23, 14)	7406	823841	119906	481979	99906	451799

Table 5. Comparison of the size of the instances generated by the direct, recursive, and compact encodings

section how $\chi_p(T_2) = 7$ can be established through automated reasoning. Proving $\chi_p(T_2) \geq 7$ is not hard, as it is enough to consider the first 9 levels of T_2 (i.e., the subgraph consisting of all vertices at distance at most 8 from the root), to obtain a sub-graph of T_2 that cannot be colored with 6 colors.

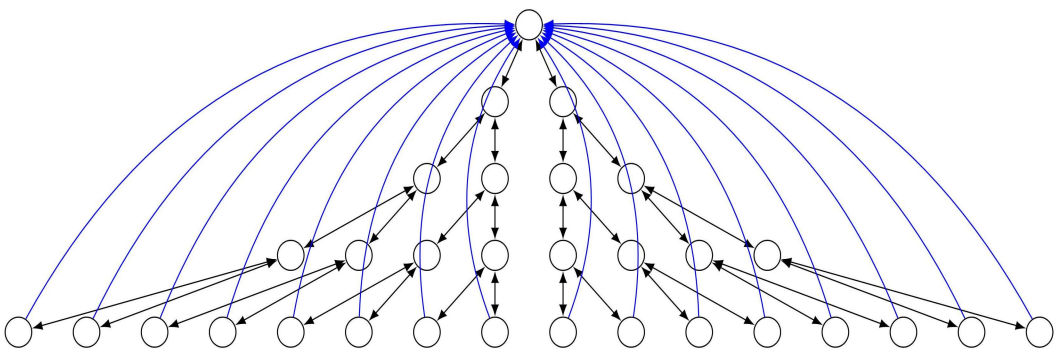


Figure 10. Illustration of the construction for $T2$ 4

This instance, under the direct encoding, is sufficiently small to be solved in less than 0.1 seconds with CaDiCaL.

However, proving $\chi_p(T_2) \leq 7$ requires a new technique. On the one hand, the upper bounds for $\chi_p(\mathbb{Z}^2)$ have been computed by adding toroidal constraints to square subgrids of $\mathbb{Z}^2$ [11]. This approach cannot be trivially replicated for the case of T_2. On the other hand, the original proof by Sloper does not use a solver, but rather a standard mathematical argument over a coloring pattern that we suspect was found manually [15].

Even though T_2 is an undirected graph, we will use a finite directed graph T'_l as a proxy to show upper bounds on $\chi_p(T_2)$. The directed graph T'_l has vertex set equal to the first l levels of T_2, directed edges in both directions between pairs of vertices (u, v) that are also connected in T_2, and finally directed edges from every leaf u to the root of the tree.The construction is illustrated in Figure 10. The reason we need directed edges instead of undirected edges is that otherwise, if two different leaves u, v were connected to the root (which is fundamental for the coloring to be extendable to T_2), they would be at distance 2, whereas their actual shortest path is the one going through their lowest common ancestor in the tree. Now, let us consider a tree T_l consisting of a root node r, from which two copies of T_l hang. We will encode a packing k-coloring for T_l in the following way.

1. Each vertex v, except for r, defines a variable $x_{v,t}$ for each color $t \in [k]$.

2. For every vertex that is not r, we create a clause stating that it has to receive at least one color, exactly as in the previous encodings.

3. Consider an isomorphism π that goes from the left copy of T'_l that hangs from r, to the right copy, and enforce now that for every vertex v in the left copy, v and $\pi(v)$ receive the same color. That is, $x_{v,t} \Leftrightarrow x\ \pi(v),t\ \forall t \in [k]$.

4. Create clauses for avoiding conflicts exactly as in the direct encoding, recalling that the distance between two vertices is now defined as the length of the shortest directed path between them.

Now, the following lemma applies the construction to upper bound $\chi_p(T_2)$.

Lemma 11 Let $I_{l,k}$ be instance resulting from the encoding described above. If $I_{l,k}$ is satisfiable, then $\chi_p(T_2) \leq k$.

Proof sketch Assume $I_{l,k}$ is satisfiable, which induces a packing k-coloring ϕ of the copies of T'_l in $I_{l,k}$. We will obtain a packing k-coloring for T_2 from ϕ. First note that T_2 can be defined recursively as a root from which 2 copies of T_2 hang. By expanding this recursive structure l times, we can say T_2 consists of a binary tree of l levels, such that from eachleaf there are 2 copies of T_2 hanging. It suffices to color the first l levels of T_2 according to ϕ and then recurse over each copy of T_2 hanging from a leaf in the l level. Now, in order to see that this coloring is actually correct we need to verify that it does not create any conflicts between the colors any pair of vertices receive. Indeed, note that the base case of the recursion cannot create any conflicts as ϕ is a valid packing k-coloring for T'_l. Then, between two copies of T_n that hang from leaves at the l^{th} level of T_n there cannot be any conflicts either, as any said conflict would have also been a conflict between the copies of T'_l in $I_{l,k}$.

6. Discussion and Future Work

Although we have managed to reduce the possible values of $\chi_p(\mathbb{Z}^2)$ to {14, 15}, determining $\chi_p(\mathbb{Z}^2)$ will probably require further techniques. We have studied the impact of different factors on the runtime of lower bounds: the basic shape of finite graphs to consider, the impact of different forms of symmetry breaking, and the cube-and-conquer parallelization approach. Proving upper bounds for this problem through local search appears to be much easier than proving lower bounds (e.g., proving the best known upper bound, 15, requires

only a few minutes of computation in a personal computer). However, local search is not able to find a solution for $D^{+}_{14,14,}$ which make us conjecture that $\chi_p(\mathbb{Z}^2) = 15$, and our future work is focused on proving this. One direction of work is studying whether the compact encoding can play a role for proving a lower bound of 15, or in finding a tiling pattern smaller than the 72 × 72 grid presented by Martin et al. [11]; although its large constant factor makes it current performance comparable to the direct encoding, part of our future work includes studying whether the same recursive principle, but under a more efficient decomposition of l_1-balls, can result in a practical speed-up. Another line of work is to compare our approach with the Linear Programming-based approach of Shao and Vesel [14]. In particular, we plan to study whether our approach can improve bounds for the packing chromatic number of distance graphs. Finally, as shown in Section 5, automated reasoning techniques are suitable for other classes of graphs as well, and thus several of the open problems presented in the survey of Brešar et al. [3] could be approached with our techniques.

References

[1] Appel, K., Haken, W. (1977). Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics, 21*(3), 429–490. https://doi.org/10.1215/ijm/1256049011

[2] Biere, A., Fazekas, K., Fleury, M., Heisinger, M. (2020). CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, & M. Suda (Eds.), *Proceedings of SAT Competition 2020 – Solver and Benchmark Descriptions* (Report No. B-2020-1, pp. 51–53). University of Helsinki.

[3] Brešar, B., Ferme, J., Klavžar, S., Rall, D. F. (2020). A survey on packing colorings. *Discussiones Mathematicae Graph Theory, 40*(4), 923. https://doi.org/10.7151/dmgt.2320

[4] Brown, S. T., Buitrago, P., Hanna, E., Sanielevici, S., Scibek, R., Nystrom, N. A. (2021). Bridges-2: A platform for rapidly-evolving and data intensive research (pp. 1–4). Association for Computing Machinery.

[5] Ekstein, J., Fiala, J., Holub, P., Lidický, B. (2010). The packing chromatic number of the square lattice is at least 12. *CoRR, abs/1003.2291*. Retrieved from https://arxiv.org/abs/1003.2291

[6] Fiala, J., Golovach, P. A. (2008). Complexity of the packing coloring problem for trees. In H. Broersma, T. Erlebach, T. Friedetzky, & D. Paulusma (Eds.), *Graph-Theoretic Concepts in Computer Science* (pp. 134–145). Springer Berlin Heidelberg.

[7] Fiala, J., Klavar, S., Lidický, B. (2009). The packing chromatic number of infinite product graphs. *European Journal of Combinatorics, 30*(5), 1101–1113. https://doi.org/10.1016/j.ejc.2008.09.014

[8] Goddard, W., Hedetniemi, S., Hedetniemi, S., Harris, J., Rall, D. (2008). Braodcast chromatic numbers of graphs. *Ars Combinatoria, 86*.

[9] Heule, M. J. H., Kullmann, O., Wieringa, S., Biere, A. (2012). Cube and conquer: Guiding CDCL SAT solvers by lookaheads. In K. Eder, J. Lourenço, & O. Shehory (Eds.), *Hardware and Software: Verification and Testing* (pp. 50–65). Springer Berlin Heidelberg.

[10] Martin, B., Raimondi, F., Chen, T., Martin, J. (2015). The packing chromatic number of the infinite square lattice is less than or equal to 16. *arXiv preprint arXiv:1510.02374v1*. Retrieved from https://arxiv.org/abs/1510.02374v1

[11] Martin, B., Raimondi, F., Chen, T., Martin, J. (2017). The packing chromatic number of the infinite square lattice is between 13 and 15. *Discrete Applied Mathematics, 225*, 136–142. https://doi.org/10.1016/j.dam.2017.03.013

[12] Neiman, D., Mackey, J., Heule, M. (2020). Tighter bounds on directed ramsey number r(7). *arXiv preprint arXiv:2011.00683*. Retrieved from https://arxiv.org/abs/2011.00683

[13] Schwenk, A. (2002). [Private communication with Wayne Goddard].

[14] Shao, Z., Vesel, A. (2015). Modeling the packing coloring problem of graphs. *Applied Mathematical Modelling, 39*(13), 3588–3595. https://doi.org/10.1016/j.apm.2014.11.060

[15] Sloper, C. (2004). An eccentric coloring of trees. *The Australasian Journal of Combinatorics, 29* [Electronic only].

[16] Soifer, A. (2016). The Hadwiger–Nelson problem. In *Springer International Publishing* (pp. 439–457). https://doi.org/10.1007/978-3-319-32162-2_14

[17] Soukal, R., Holub, P. (2010). A note on packing chromatic number of the square lattice. *The Electronic Journal of Combinatorics, 17*(1). https://doi.org/10.37236/466