EPiC Series in Computing

Volume 94, 2023, Pages 386–404

Proceedings of 24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning

Toward Optimal Radio Colorings of Hypercubes via SAT-solving

Bernardo Subercaseaux and Marijn J.H. Heule

Carnegie Mellon University
{bsuberca, mheule}@cs.cmu.edu

Abstract

Radio 2-colorings of graphs are a generalization of vertex colorings motivated by the problem of assigning frequency channels in radio networks. In a radio 2-coloring of a graph, vertices are assigned integer colors so that the color of two vertices u and v differ by at least 2 if u and v are neighbors, and by at least 1 if u and v have a common neighbor. Our work improves the best-known bounds for optimal radio 2-colorings of small hypercube graphs, a combinatorial problem that has received significant attention in the past. We do so by using automated reasoning techniques such as symmetry breaking and Cube and Conquer, obtaining that for $n = 7$ and $n = 8$, the coding-theory upper bounds of Whittlesey et al. (1995) are not tight. Moreover, we prove the answer for $n = 7$ to be either 12 or 13, thus making a substantial step towards answering an open problem by Knuth (2015). Finally, we include several combinatorial observations that might be useful for further progress, while also arguing that fully determining the answer for $n = 7$ will require new techniques.

1 Introduction

Frequency Assignment Problems (FAPs) in radio networks, are mathematical formulations for the problem of assigning frequencies to radio transmitters in a way that maximizes frequency reuse while keeping signal interference to acceptable levels [1, 2, 3, 4]. In 1980, Hale formalized FAPs as graph labeling problems, initiating an active area of combinatorial research [2, 3, 5, 6]. Arguably the most elemental formulation of a FAP is that of radio 2-colorings, introduced by Griggs and Yeh in the early 1990s [7], which we define next.

Definition 1 (Radio 2-coloring)**.** *Given a graph $G = (V, E)$, and a natural number $s \geq 1$, a radio 2-coloring of G of span s is a function $f : V(G) \to \{0, \ldots, s\}$[1] such that:*

1. *$f(u) \neq f(v)$, if u and v are at distance 2.*
2. *$|f(u) - f(v)| \geq 2$, if u and v are at distance 1.*

Naturally, this definition induces a notion of chromatic number:

[1] One can equivalently consider labels in $\{1, \ldots, s+1\}$, but we stick to the range $\{0, \ldots, s\}$ for consistency with the existing literature [5, 6].

Definition 2 (Radio 2-chromatic number)**.** *The radio 2-chromatic number of a graph G, denoted by $\lambda(G)$, is the smallest s for which G admits a radio 2-coloring with span s.*[2]

The radio 2-chromatic number has been studied for several graph families [5, 6], and in particular, several bounds are known for the family of *hypercube graphs* [7, 9, 10].

Definition 3. *For any natural number $n \geq 1$, the hypercube graph of order n, denoted by Q_n, has vertex set $\{0,1\}^n$, and edges between elements of $\{0,1\}^n$ that differ in exactly one component.*

Example 1. *Figure 2 (on page 5) illustrates a radio 2-coloring for Q_4 with 8 colors (span of 7), which turns out to be optimal [10].*

In the early 1990s, Griggs and Yeh [7], and Jonas [11] proved the first general bounds on $\lambda(Q_n)$, and later on Whittlesey et al. [10] improved the general upper bound by 1, while also providing a different and more refined upper bound. The following theorems summarize these results.

Theorem 1 (Griggs and Yeh [7], Jonas [11])**.** *For any $n \geq 5$, we have $n+3 \leq \lambda(Q_n) \leq 2n+1$.*

Theorem 2 (Whittlesey et al. [10])**.** *For any $n \geq 1$, we have $\lambda(Q_n) \leq 2n$.*

Theorem 3 (Whittlesey et al. [10])**.** *For any $k \geq 1$ and $q \leq k+1$, we have*

$$\lambda(Q_{2^k-q}) \leq 2^k + 2^{k-q+1} - 2.$$

Table 1 shows the upper bounds obtained through Theorem 3 for small values of n, while Figure 1 shows the general curve of both upper and lower bounds.

Interestingly, it turns out that computing $\lambda(Q_n)$ is a remarkably hard computational problem, even for very modest values of n. In *The Art of Computer Programming* [12], Donald Knuth presented the following open problem:

Open Problem 1. *Find $\lambda(Q_n)$ for any $n > 6$.*[3]

To the best of our knowledge, all previous conjectures about the value $\lambda(Q_n)$ have been shown to fail, as we summarize next. According to Calamoneri [5], Griggs and Yeh conjectured that $\lambda(Q_n) = n+3$ (i.e., the lower bound from Theorem 1), however, this is incompatible with Griggs and Yeh's 1992 article which mentions already that $\lambda(Q_8) \geq 12$, and that *no evident pattern* emerges from the values $\lambda(Q_n)$ with $n \leq 5$ that they computed [7]. On the other hand, Frieder et al. [9] proposed an algorithm for radio 2-coloring Q_n, which they incorrectly conjectured to be optimal. However their algorithm yields a radio 2-coloring of span 15 for Q_5, whereas it is known that $\lambda(Q_5) = 8$ [7]. The last candidate formula is therefore the upper bound from Theorem 3, which matches the answer up to $n = 6$. However, this article proves that this cannot be a formula for $\lambda(Q_n)$ either. Indeed, we show the following.

Theorem 4. $12 \leq \lambda(Q_7) \leq 13$ *and* $12 \leq \lambda(Q_8) \leq 14$.

[2] Unfortunately, the literature on this problem does not have a uniform notation; radio 2-colorings are also known as $L_1(2,1)$-labelings [7], $L(2,1)$-labelings [6], or $L_{(2,1)}$-labelings [8], and the radio-2-chromatic number is sometimes denoted as $\lambda(\cdot)$ [7], $\lambda_2(\cdot)$ [6], $\lambda_{2,1}(\cdot)$ [5], or $\mathrm{rc}_2(\cdot)$ [2].

[3] It is worth mentioning that Open Problem 1 is assigned an estimated difficulty of 46 in Knuth's scale, which ranges from 0 to 50.

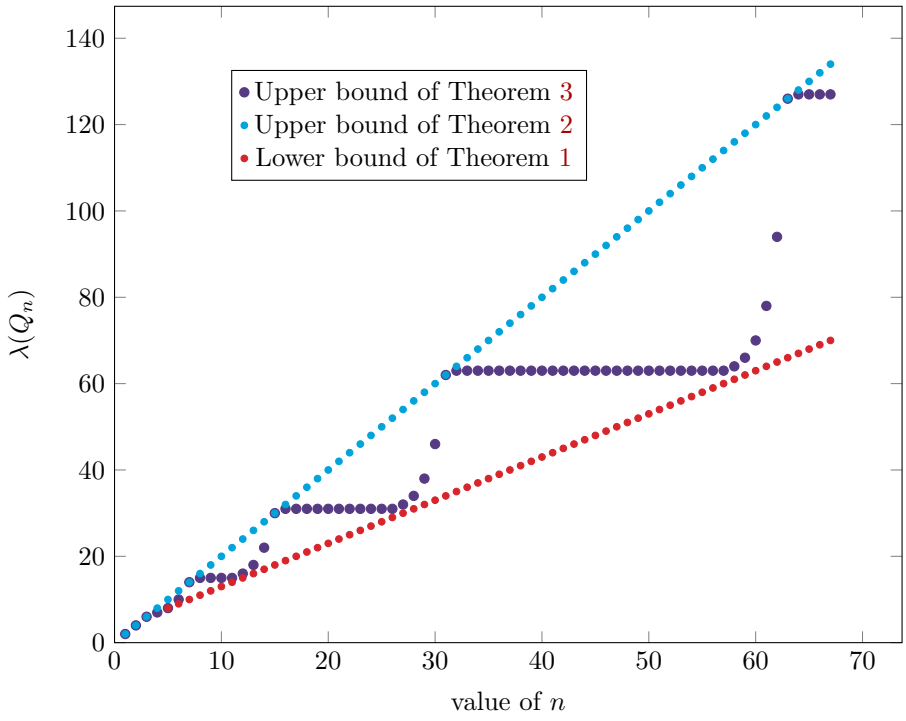

Figure 1: Illustration of the bounds obtained by Theorems 1 to 3.

As the upper bounds obtained from Theorem 3 are 14 for $n = 7$ and 15 for $n = 8$, we show the first instance of a gap between $\lambda(Q_n)$ and Theorem 3. Moreover, in Section 5 we show evidence suggesting that the entire coding theoretical approach of Whittlesey et al. [10] cannot directly provide a better bound than 14 for $\lambda(Q_7)$, thus implying that new techniques will be required. The following paragraphs present more context on the computation of $\lambda(Q_n)$.

Asymptotics of $\lambda(Q_n)$. The asymptotic behavior of $\lambda(Q_n)$ is not well understood yet; Whittlesey et al. [10] proved that $\liminf \lambda(Q_n)/n = 1$, but to the best of our knowledge, it is not known whether $\limsup \lambda(Q_n)/n < 2$. As a consequence, improving computational methods to compute $\lambda(Q_n)$ for slightly larger values of n might serve as a guide toward understanding its limiting behavior.

Complexity of computing $\lambda(Q_n)$. The problem of deciding whether $\lambda(G) \leq s$ for an arbitrary graph G was proven to be NP-hard by Griggs and Yeh [7]. Moreover, Fiala et al. [13] proved that this problem is hard for every fixed $s \geq 4$. Nonetheless, particular classes of graphs might make the problem easier. For example, Chang and Kuo give a polynomial time algorithm for trees [4]. To the best of our knowledge, no polynomial time algorithm is known for hypercubes. The class of regular graphs, to which hypercubes belong, is enough for NP-hardness, as proved by Fiala and Kratochvíl [8]. Moreover, Fiala et al. [14] showed intractability in several fixed-parameter settings, while showing efficient algorithms for graphs of bounded vertex-cover number or *neighborhood diversity*, both of which are unbounded parameters for the class of hypercubes.

Table 1: Best known bounds on $\lambda(Q_n)$. Bold values show our improvements, and italic values show the improvements made by Jonas [11] over the general bound of Theorem 1.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
upper bound	2	4	6	7	8	10	**13**	**14**	15	15	15	16	18	22	30	31
lower bound	2	4	6	7	8	10	**12**	*12*	12	13	14	15	16	17	18	*20*

SAT-solving to compute $\lambda(Q_n)$. Our work is primarily concerned with computing $\lambda(Q_n)$ exactly for small values of n, hopefully contributing to reveal patterns that could help a sharper understanding of $\lambda(Q_n)$ as a general function of n. Given the hardness results summarized in the previous paragraph, we do not expect efficient algorithms for deciding whether $\lambda(Q_n) \leq s$, for given values of n and s. Therefore, our approach consists of solving SAT instances [15] that encode whether $\lambda(Q_n) \leq s$. In order to make them tractable, we perform several optimizations over naively generated instances, such as *symmetry breaking* [15] and *Cube and Conquer* [16]. Our work follows the line of the approach of Subercaseaux and Heule to compute the packing-chromatic number of the infinite square grid [17, 18].

We conclude this introduction with the organization of the paper. Section 2 presents the direct encoding of $\lambda(Q_n) \leq s$ as a CNF formula $\mathcal{Q}_n^{\leq s}$. Then, Section 3 describes the symmetries in $\mathcal{Q}_n^{\leq s}$ and how to break them in order to reduce solving time. Section 4 shows how the Cube and Conquer paradigm can be used on this problem, and Section 5 shows how the coding theoretical approach of Whittlesey et al., [10] can be recreated with SAT solving and shown to be tight. Then, Section 6 studies how from the combinatorial proofs of lower bounds e.g., Theorem 1) one can derive additional constraints that speed up SAT solving. Finally, Section 7 details our experimental results, and Section 8 presents concluding remarks and challenges to guide further progress.

2 Encoding

Let $\mathcal{Q}_n^{\leq s}$ be the CNF encoding we will build for the problem of determining if $\lambda(Q_n) \leq s$. To construct $\mathcal{Q}_n^{\leq s}$ start by defining variables $x_{u,c}$ meaning that vertex u receives color c (i.e., $f(u) = c$). These are the only variables in $\mathcal{Q}_n^{\leq s}$, and thus we have $\#\text{vars}(\mathcal{Q}_n^{\leq s}) = 2^n \cdot (s+1)$.

Next, for every vertex u, add a clause to $\mathcal{Q}_n^{\leq s}$ forcing that u will be colored:

$$\bigvee_{c \in \{0,\ldots,s\}} x_{u,c}.$$

Then, for each pair of vertices u, v at distance 2 from each other, and every color $c \in \{0, \ldots, s\}$, add a clause

$$\overline{x_{u,c}} \vee \overline{x_{v,c}}.$$

Finally, for every pair of colors $c_1, c_2 \in \{0, \ldots, s\}$ with $|c_1 - c_2| \leq 1$, and pair of adjacent vertices u, v, add to $\mathcal{Q}_n^{\leq s}$ the clauses

$$\overline{x_{u,c_1}} \vee \overline{x_{v,c_2}}.$$

This is enough for constructing $\mathcal{Q}_n^{\leq s}$, and by simply counting the described clauses we obtain:

$$\#\text{clauses}(\mathcal{Q}_n^{\leq s}) = 2^n + (s+1)2^{n-1}\binom{n}{2} + n2^n\,(3s+1)\,.$$

Table 2: Size of instances $\mathcal{Q}_n^{\leq s}$, for $n \leq 8$, with s chosen as the upper bound for $\lambda(Q_n)$ according to Theorem 3.

n	s	#vars	#clauses
1	2	6	16
2	4	20	118
3	6	56	548
4	7	128	1808
5	8	288	5472
6	10	704	17248
7	14	1920	58816
8	15	4096	151808

Table 2 displays the number of variables and clauses for some instances of interest.

3 Symmetry Breaking

In the context of SAT solving one can distinguish between two kinds of symmetries: *internal symmetries*, which are symmetries within a given solution, and *solution symmetries*, which

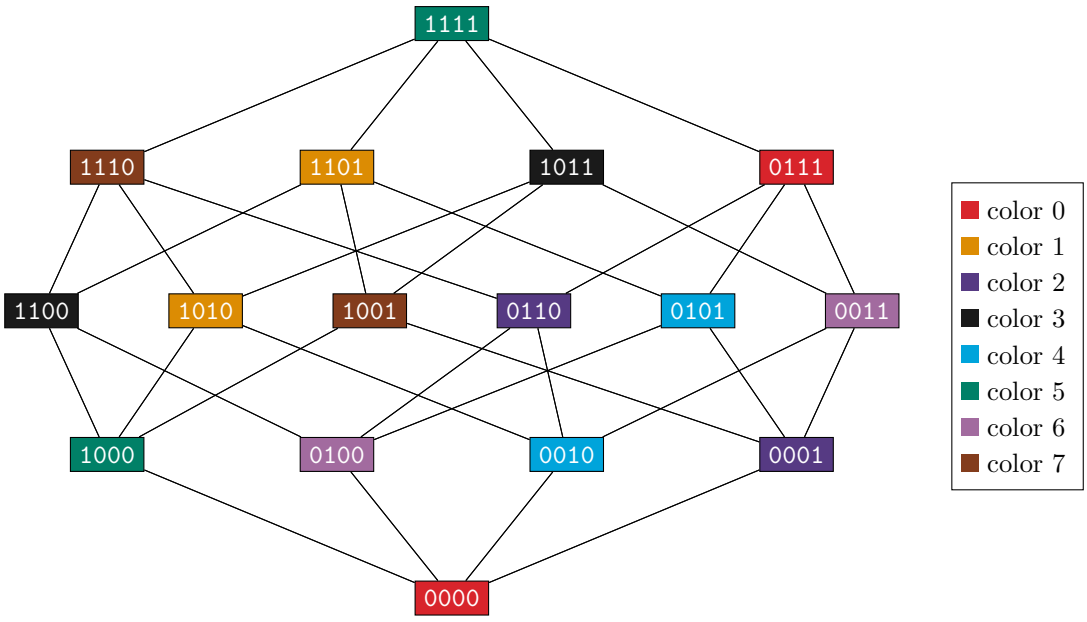

Figure 2: Illustration of a radio 2-coloring of Q_4 (represented by its Hasse diagram) with 8 colors. The span of this coloring is 7, thus proving that $\lambda(Q_4) \leq 7$. Interestingly, this solution is highly symmetric: for any vertex v it holds that $f(v \oplus \texttt{0111}) = f(v)$, where $\oplus$ denotes the bitwise-xor, and moreover, $f(v \oplus \texttt{1001}) = s - f(v)$. Section 3 expands on these forms of symmetries within solutions, and how they can be exploited to speed up computation.

are symmetries between different solutions [15, 19]. In other words, internal symmetries map particular solutions of combinatorial problems onto themselves, while solution symmetries map solutions onto different (but equivalent) ones. In general, internal symmetries are useful for obtaining faster SAT results, while solution symmetries allow for obtaining UNSAT results faster.

For a simple example, if we consider the problem of finding pairs of integers a, b such that $a + b = 10$, then the function $g(a, b) = (a - 7, b + 7)$ is a map between solutions (e.g., $(6, 4) \to (-1, 11)$), whereas the particular solution $(5, 5)$ is invariant under the symmetry $h(a, b) = (b, a)$, and thus we say it contains an internal symmetry. To exploit the solution symmetry g, we can enforce, for example, the constraint $a < 0$, as any solution (a, b) can be mapped onto one with $a < 0$ by successive applications of g. The idea is that, in a problem where one searches for an UNSAT proof, the addition of extra constraints over the solution will make it easier for the solver to deduce that no solution to the problem exists. The point of internal symmetries, on the other side, is that, if we know that there exists at least one solution respecting a given internal symmetry, we can reduce the search space by enforcing the symmetry as a constraint. Continuing with our example, by knowing that at least one solution respecting the symmetry h exists, we can enforce the constraint $a = b$, thus reducing the problem to finding a single integer a such that $2a = 10$.

We next describe both solution symmetries and internal symmetries over $\mathcal{Q}_n^{\leq s}$. While it is simple to observe that if f is a radio 2-coloring with span s for any graph G, then $f'(v) := s - v$ is also a radio 2-coloring, it turns out there are many more symmetries between solutions for instances $\mathcal{Q}_n^{\leq s}$; as many as $n! \cdot 2^n$. To understand all such symmetries, we first describe the automorphisms of Q_n, that is, the symmetries of hypercube graphs, and then explain how to use them for breaking symmetry in $\mathcal{Q}_n^{\leq s}$.

Automorphisms of Q_n. It is well known that the group of automorphisms of Q_n (also known as the hyperoctahedral group) has size $n! \cdot 2^n$. We will show this in a way that is helpful for guiding our symmetry breaking approach. Let u be any vertex of Q_n, and define the closed neighborhood of u, denoted by $\mathcal{N}[u]$, as

$$\mathcal{N}[u] \coloneqq \{u\} \cup \mathcal{N}(u),$$

where $\mathcal{N}(u)$ denotes the set of neighbors of u in Q_n. The key for understanding the automorphisms of Q_n is that any such automorphism ϕ is uniquely determined by

$$F(u) \coloneqq \{(v, \phi(v)) \mid v \in \mathcal{N}[u]\},$$

the mapping of the closed neighborhood of u. This is because, as we prove next, every vertex w that is not in the set $\mathcal{N}[u]$, is mapped to a vertex $\phi(w)$ that is uniquely determined by its distance to the elements of $\mathcal{N}[u]$. Indeed, given that Q_n is clearly a vertex-transitive graph, we can assume without loss of generality that $u = \vec{0}$ and $\mathcal{N}(u)$ is the set of canonical vectors $\mathbf{e}_i$ (i.e., the vector with a 1 in its i-th, and 0 everywhere else), for $0 \leq i < n$. If we let b be the distance between $u = \vec{0}$ and another vertex w, then w has exactly b bits on, and moreover, its i-th bit is on if and only if the distance between w and $\mathbf{e}_i$ is $b - 1$. Therefore, the distances between a vertex w and the vertices in $\mathcal{N}[u]$ are enough to fully specify the identity of w. Finally, the number of choices for $F(u)$ can be computed as follows: there are 2^n options for $\phi(u)$, as the graph is vertex-transitive, and once $\phi(u)$ has been determined, the automorphism condition implies that $\phi(v) \in \mathcal{N}(\phi(u))$ for every $v \in \mathcal{N}(u)$. But this is equivalent to saying that ϕ needs to bijectively map the set $\bigcup_{v \in \mathcal{N}(u)} \{\phi(v)\}$ to $\mathcal{N}(\phi(u))$, which can be done in $n!$ different ways

as each set has size n.

In order to obtain an $n! \cdot 2^n$ factor of improvement, we need to break all symmetries of Q_n, which means that we need to distinguish a particular vertex u, obtaining a 2^n factor, and then break the symmetry on the neighborhood of u for the $n!$ factor.

Symmetry breaking I. First, given that Q_n is vertex-transitive, we can assume that if any vertex receives a given color c, then vertex $\vec{0}$ receives color c. This means that if in radio 2-coloring of Q_n there are ℓ vertices getting color c, we obtain an improvement factor of $\frac{\ell}{2^n}$. If $\ell = 1$ this is clearly optimal, but for larger values of ℓ a more sophisticated approach that we detail next is required.

For any vertex v, let us define $S(v)$ as the sorted sequence of colors that the vertices in $\mathcal{N}(v)$ receive. Then, we can break the symmetry between the ℓ vertices receiving color c by assuming that $S(\vec{0})$ is lexicographically smallest amongst the sequences $S(v)$ for every vertex v that receives color c. In case there are no two vertices u, v receiving color c such that $S(u) = S(v)$, this assumption divides the number of cases by ℓ and thus fully breaks symmetries of Q_n. Enforcing $S(\vec{0})$ to be lexicographically smallest can be expensive in terms of the number of clauses, and thus we do not implement such a further optimization. Given that a priori we cannot assume that any specific color c will be assigned to some vertex in an optimal radio 2-coloring (besides color 0, which can easily be seen to always appear in any optimal radio 2-coloring without loss of generality), we use the following conditional constraint: *if at least one vertex receives color c, then vertex $\vec{0}$ receives color c.* Therefore, we add the following $2^n - 1$ clauses:

$$x_{\vec{0},c} \vee \overline{x_{u,c}}, \quad \forall u \in \{0,1\}^n \setminus \{\vec{0}\}.$$

Symmetry breaking II. Once a vertex u (e.g., $\vec{0}$) has been distinguished, breaking the symmetry of its neighborhood $\mathcal{N}(u)$ can be done as follows. First, observing that all vertices in $\mathcal{N}(u)$ are at distance at most 2 from each other, we know they must receive different colors. Furthermore, we can assume the sequence of colors they receive will be strictly increasing.

To implement this, consider without loss of generality that $u = \vec{0}$, it is enough to enforce that if $\mathbf{e}_i$, the unit vector with a 1 in its i-th position, gets some color a, then $\mathbf{e}_{i+1}$ cannot get any color $b < a$. Concretely, we add the following $O(ns^2)$ clauses:

$$\overline{x_{\mathbf{e}_i,a}} \vee \overline{x_{\mathbf{e}_{i+1},b}}, \quad \forall i \in \{1,\ldots,n-1\}, a \in \{0,\ldots,s\}, b \in \{0,\ldots,a-1\}.$$

This implies that the n different colors assigned to vertices in $\mathcal{N}(\vec{0})$ cannot be permuted anymore, thus dividing by $n!$, the number of permutations over the colors assigned to $\mathcal{N}(\vec{0})$.

Internal symmetries. Internal symmetries can be observed in optimal radio 2-colorings. For example, the coloring of span 7 presented in Figure 2 has the following internal symmetry:

> *Each color $c \in \{0,\ldots,7\}$ is assigned to exactly two vertices, which we denote u_c and v_c. It then holds, for every $c \in \{0,\ldots,7\}$, that $u_c = v_c \oplus \texttt{0111}$, where $\oplus$ denotes the bitwise-xor, interpreting the vertices of $\{0,1\}^7$ as bit-vectors.*

For example, color 6 is assigned to vertices `0011` and `0100`, while color 3 is assigned to vertices `1011` and `1100`. The knowledge of this internal symmetry allows for a speed-up by simply adding constraints stating that any pair of vertices u, v such that $u = v \oplus \texttt{0111}$ must

receive the same color. Unfortunately, it is not the case that for every n such that Q_n admits a solution of span s, there exists a mask m and a solution symmetric under $u = v \oplus m$ for all vertices. A slightly more involved internal symmetry that appears to exist for every optimal span (although we leave a proof or refutation of its existence as a challenge in Section 8) is the following (for every n):

> *There exists a mask m, such that at least one radio 2-coloring f for Q_n of optimal span s exists which satisfies that: $f(v \oplus m) = s - f(v)$ for every vertex v.*

This internal symmetry is also present in Figure 2 using mask `1001`. In Appendix A.1 and Appendix A.2, we present certificates for our improved upper bounds for $n = 7$ and $n = 8$, using the same internal symmetry although with different masks.

Interestingly, finding a mask m that works for a particular instance $\mathcal{Q}_n^{\leq s}$ is not as hard as it can appear at first. Naively, the total number of masks is 2^n, and testing each candidate mask m_i can be done by setting a timeout parameter t and running the instance $\mathcal{Q}_n^{\leq s}$, with the additional constraints corresponding to the candidate mask m_i, for that amount of time. Such a process would take time $2^n \cdot t$ in the worst case. However, by observing as in Section 3 that the different bits are interchangeable under permutations, we can assume that the bits of the desired mask m, if it exists, are sorted. This way, one only needs to test n different candidate masks, reducing the time down to $t \cdot n$.

4 Cube and Conquer

The Cube and Conquer [16] approach to SAT solving consists of *splitting* a SAT formula φ into a sequence of formulas $\varphi_1, \ldots, \varphi_m$, in such a way that φ is satisfiable if, and only if, at least one of the formulas φ_i is satisfiable. The main consequence is that it is then possible for a solver to work on the different instances φ_i in parallel. Moreover, *incremental solvers* can take advantage of the overlap between the different φ_i formulas, reusing computation across them [16].

We now present a general description of the Cube and Conquer approach, and then a particular instantiation of it for the problem at hand.

The general paradigm. If $\psi = (c_1 \vee c_2 \vee \cdots \vee c_m)$ is a tautological DNF, then we have

$$\text{SAT}(\varphi) \iff \text{SAT}(\varphi \wedge \psi) \iff \text{SAT}\left(\bigvee_{i=1}^{m} (\varphi \wedge c_i)\right) \iff \text{SAT}\left(\bigvee_{i=1}^{m} \varphi_i\right),$$

where the different $\varphi_i \coloneqq (\varphi \wedge c_i)$ are the instances resulting from the split.

Intuitively, each cube c_i represents a *case*, i.e., an assumption about a satisfying assignment to φ, and soundness comes from ψ being a tautology, which means that the split into cases is exhaustive. If the split is well designed, then each φ_i is a particular case that is substantially easier to solve than φ, and thus solving them all in parallel can give significant speed-ups, especially considering the sequential nature of CDCL, at the core of most solvers.

Split algorithms for radio coloring hypercubes. As is generally the case in parallel computation, optimal performance requires all cores or processors to be equally busy. As a consequence, we want the different formulas φ_i to correspond to cases of roughly the same difficulty. Moreover, the number m of cases in the split should be fairly large, as that way the

total work associated with all cases is closer to being a continuous variable that could be divided exactly by p, the number of processors. Concrete examples of the impact of these parameters on solving time can be found in recent work by Subercaseaux and Heule using the Cube and Conquer paradigm on a different coloring problem [17, 18]. Furthermore, the `march_cc` tool [16] is able to automatically generate splits, even allowing the user to specify the target number of cases. However, due to its generality, `march_cc` can be beaten on specific problems by designing split algorithms with knowledge of the problem at hand.

We now present a concrete split algorithm for this problem. A comparison of the performance between our algorithm and `march_cc` is discussed in Section 7.

A Custom Split Algorithm. Given our symmetry breaking approach assigns a color c to vertex $\vec{0}$ (see Section 3), this algorithm assigns colors to the vertices in B_2^n the set of vertices of $\{0,1\}^n$ with exactly 2 bits on. The motivation for this is that all such vertices are at distance 2 from $\vec{0}$ and thus cannot share its color c. Let $\mathcal{B} \subseteq B_2^n$ be a subset of B_2^n that will be chosen manually. Then, for each assignment $q : \mathcal{B} \to \{0, \dots, s\} \setminus \{c\}$, we create a cube

$$c_q := \bigwedge_{v \in \mathcal{B}} x_{v,q(v)}.$$

This algorithm would generate $s^{|\mathcal{B}|}$ cubes a priori. However, some of these cubes are clearly unsatisfiable, as if $u, v \in B_2^n$ share a bit on (e.g., `0101` and `1001`), then u and v are distance 2 and thus cannot be assigned the same color. This allows us to immediately discard any assignment q such that $q(u) = q(v)$. By doing so, we significantly reduce the number of cubes.

5 A Coding Theoretical Encoding

Considering that the best known theoretical upper bounds for $\lambda(Q_n)$ use coding theory methods [10], we studied whether a coding theoretical encoding allows SAT solvers to match, or improve Theorem 3. Our main result in this section is that, at least for $n = 7$, the coding theory approach of [10] cannot be trivially modified to obtain a better bound. Amongst other things, coding theory studies how binary messages, say from $\{0,1\}^k$, can be exchanged in a way that tolerates noise in the communication channel [20]. More precisely, a binary word $w \in \{0,1\}^k$ can be encoded into a codeword $c_w \in \{0,1\}^n$, with $n > k$, such that flipping a certain number of bits in c_w (say, at most 2) still allows the recipient to recover the original word w. Consider a code $C : \{0,1\}^k \to \{0,1\}^n$, and let I be its image. Naturally, being able to decode codewords with at most 2 bits flipped implies that any pair of codewords $c_{w_1}, c_{w_2} \in I$ must differ by at least 5 bits (if c_{w_1} differs from c_{w_2} in 4 bits, there is a distorted word $\tilde{c}$ that differs both from c_{w_1} and c_{w_2} in 2 bits, and thus a recipient would not be able to tell whether $\tilde{c}$ corresponds to a distortion on c_{w_1} or on c_{w_2}).

This problem is tightly related to radio 2-colorings; Whittlesey et al. [10] use a linear code to map Q_n into Q_k $(k < n)$ in such a way that vertices u, v of Q_n that are at distance 2 from each other are mapped onto different vertices of Q_k, and then they color Q_k injectively. For example, to obtain an upper bound of 14 for $\lambda(Q_7)$, they map Q_7 to Q_3, and then label Q_3 using colors $\{0, \dots, 14\}$. A natural question, considering that Q_3 only has 8 vertices, is whether after the mapping one could obtain a radio 2-coloring by using colors $\{0, \dots, 13\}$ for Q_3. We show this to not be possible by using the following encoding.

Let n and k be fixed. Then, create variables $x_{u,v}$ for $u \in \{0,1\}^n, v \in \{0,1\}^k$ that represent that vertex u will be mapped to vertex v. Naturally, we create next 2^n clauses stating that

each vertex in Q_n must be mapped onto some vertex of Q_k:

$$\forall u \in \{0,1\}^n, \quad \bigvee_{v \in \{0,1\}^k} x_{u,v}.$$

It is not necessary to force that each vertex of Q_n will be mapped onto a single vertex of Q_k. Then, create variables $y_{v,c}$ stating that vertex $v \in Q_k$ will receive color c. We add therefore 2^k clauses:

$$\forall v \in \{0,1\}^k, \quad \bigvee_{c \in \{0,\ldots,s\}} y_{v,c}.$$

We then must enforce that if neighboring vertices u_1, u_2 from Q_n are mapped onto v_1, v_2 respectively, then the colors received by v_1 and v_2 must differ by 2. Similarly, if u_1 and u_2 were at distance 2, then v_1 and v_2 must receive different colors. Concretely, we add the following clauses corresponding to the distance 1 condition:

$$\begin{aligned}
&\forall u_1, u_2 \in \{0,1\}^n \text{ such that } \text{distance}(u_1, u_2) = 1, \\
&\quad \forall c_1, c_2 \in \{0, \ldots, s\} \text{ such that } |c_1 - c_2| \leq 1, \\
&\quad\quad \forall v_1, v_2 \in \{0,1\}^k, \\
&\quad\quad\quad \left(\overline{x_{u_1,v_1}} \vee \overline{x_{u_2,v_2}} \vee \overline{y_{v_1,c_1}} \vee \overline{y_{v_2,c_2}}\right).
\end{aligned}$$

Similarly, the following clauses take care of the distance 2 condition:

$$\begin{aligned}
&\forall u_1, u_2 \in \{0,1\}^n \text{ such that } \text{distance}(u_1, u_2) = 2, \\
&\quad \forall c \in \{0, \ldots, s\}, \\
&\quad\quad \forall v_1, v_2 \in \{0,1\}^k, \\
&\quad\quad\quad \left(\overline{x_{u_1,v_1}} \vee \overline{x_{u_2,v_2}} \vee \overline{y_{v_1,c}} \vee \overline{y_{v_2,c}}\right).
\end{aligned}$$

Although the size of this encoding is extremely large, instances for $n = 7$ are still solvable. Concretely, it amounts to

$$\Theta\left(s \cdot n^2 \cdot 2^{2k+n}\right)$$

many clauses, and yet by running it with parameters $n = 7, k = 3$ and $s = 14$ we quickly recover the solution of span 14 from Whittlesey et al. from [10]. Furthermore, by obtaining an UNSAT result for parameters $n = 7, k = 3$ and $s = 13$, we confirm that no mapping from Q_7 to Q_3 allows for a solution of span 13. In contrast, we show in Theorem 4 that there is a more general solution of span 13 for Q_7, implying therefore that the coding theory approach of [10] is inherently sub-optimal, as opposed to the possibility of their upper bound being algebraically loose. This computation took 438 seconds (hardware details in Section 7).

6 Lower Bound Optimizations

Even though Theorem 1 establishes that $\lambda(Q_n) \geq n + 3$, and the proof of said bound is quite simple, SAT solvers struggle to prove for example that $\lambda(Q_{11}) \geq 14$, even taking over 2 seconds to deduce that $\lambda(Q_{12}) \geq 14$, which is almost a trivial result. This section shows how we can

take inspiration from the general combinatorial proofs of these lower bounds to speed up SAT solvers.

First, let us consider a proof of $\lambda(Q_n) \geq n+2$, which is slightly simpler than the bound of Theorem 1.

Fact 1. *For $n \geq 2$, $\lambda(Q_n) \geq n+2$.*

Rather than providing a formal proof here (which we offer in Appendix A.3 for completeness), let us provide a slightly incomplete argument that has the benefit of making our optimization more intuitive.

Assume the symmetry-breaking predicates described in Section 3 have been applied, and vertex $\vec{0}$ is assigned color 3, for example. Then, given that the sequence of colors for the neighbors of $\vec{0}$ is strictly increasing, the best case (i.e., minimizing the span) for that sequence of n colors is

$$0, 1, 5, 6, \ldots, n, n+1, n+2,$$

from which immediately follows that $\lambda(Q_n) \geq n+2$. However, we observe experimentally that SAT solvers do not obtain this conclusion succinctly without additional guidance. We can provide such guidance by adding clauses of the form:

$$\left(\bigvee_{c=0}^{m-1} x_{\mathbf{e}_i, c}\right) \vee \overline{x_{\mathbf{e}_{i+1}, m}}, \quad \forall i \in \{1, \ldots, n-1\}, \forall m \in \{0, \ldots, s\},$$

recalling that $\mathbf{e}_i$ denotes the bit-vector with a single bit on, in the i-th position. The reason we can add these clauses without losing any solution is that, if the i-th neighbor of $\vec{0}$ does not receive any color in $\{0, \ldots, m-1\}$ then it must receive a color greater or equal then m, and thus the $(i+1)$-th neighbor of $\vec{0}$ cannot receive color m given that the sequence of colors must be increasing. Furthermore, the addition of these clauses greatly simplifies the reasoning required to deduce $\lambda(Q_n) \geq n+2$; if $\mathbf{e}_1$ does not get color 0, then $\mathbf{e}_2$ cannot get color 1, and given it also cannot get colors in $\{2, 3, 4\}$, it must get color at least 5. By continuing this simple line of deduction we infer that $\mathbf{e}_n$ must get color at least $n+2$. The addition of these clauses brings down the runtime for instance $\lambda(Q_{12}) \leq 13$ from 2.5 seconds to 1.2 seconds. More importantly, without these additional clauses, the solver learns 21142 clauses, faces 21693 conflicts, and performs 5390207 propagations. On the other hand with the additional clauses, it only performs 250 propagations and faces exactly 1 conflict. Thus, the 1.2 seconds of solving time are almost entirely used in parsing the clauses. The proof for the lower bound in Theorem 1 is slightly more involved, as it uses a counting argument over the vertices at distance 2 from $\vec{0}$. We leave the task of designing additional clauses that could guide solvers towards such a proof as future work, and discuss it further in Section 8.

7 Experiments

Experimental Setup. In terms of software, for CNF instances we run experiments both on the state-of-the-art solver `CaDiCaL` [21], and on `YalSAT/PalSAT` [22]. Experimentally, local search solvers (e.g., `YalSAT/PalSAT`) worked much better than CDCL solvers (e.g., `CaDiCaL`) for satisfiable instances. All Cube and Conquer experiments were run using a new implementation of parallel `iCaDiCaL` because it supports incremental solving [16, 23]. In terms of hardware, all our experiments were run on the Bridges2 cluster of the Pittsburgh Supercomputing Center [24], which has 512GB of RAM and two AMD EPYC 7742 CPUs, each with: 64 cores of 2.25-3.40GHz, 256MB L3, and 8 memory channels.

Table 3: Runtime in seconds for instances $\mathcal{Q}_6^{\leq 9}$ (1 core with `CaDiCaL`) and $\mathcal{Q}_7^{\leq 11}$ (128 cores with `iCaDiCaL`) as a function of the color assigned to vertex $\vec{0}$. The best time for each instance is written in boldface.

Instance	Assigned color											
	0	1	2	3	4	5	6	7	8	9	10	11
$\mathcal{Q}_6^{\leq 9}$	192	165	130	144	**114**	116	157	122	168	186	-	-
$\mathcal{Q}_7^{\leq 11}$	4853	3974	2977	3337	3248	3096	3669	3370	2799	**2688**	3524	4356

Main Theorem. For proving the first part of Theorem 4, we need to obtain a satisfying assignment to $\mathcal{Q}_7^{\leq 13}$. A solution for this instance is obtained in 11.67 seconds with `PalSAT` (128 cores), and is depicted in Figure 3. For the lower bound, we need an UNSAT result for $\mathcal{Q}_7^{\leq 11}$, which turns out to be significantly harder than the upper bound computation. Our fastest run took 2588 seconds using our custom split algorithm for Cube and Conquer, as well as the symmetry breaking predicates of type I and II described in Section 3, forcing vertex $\vec{0}$ to receive color 2[4]. On the other hand, by exploiting internal symmetry as described in Section 3, we obtain a satisfying assignment to $\mathcal{Q}_8^{\leq 14}$ in 1.93 seconds with `PalSAT` (128 cores). In contrast, when removing the internal symmetry constraints, the runtime goes up to 316.01 seconds. This provides the second upper bound, while the lower bound comes simply from using that $\lambda(Q_7) \leq \lambda(Q_8)$.

7.1 Symmetry Breaking Experiments

We experiment with different parameters and heuristics for doing symmetry breaking. First, considering that in principle the color c assigned to vertex $\vec{0}$ (see Section 3) can take any value in $\{0, \ldots, k\}$, we experimented with all choices of c over instances $\mathcal{Q}_6^{\leq 9}$ and $\mathcal{Q}_7^{\leq 11}$, both of which are unsatisfiable. The results are displayed in Table 3 and Figure 4. For the instance $\mathcal{Q}_6^{\leq 9}$, only symmetry breaking predicates of type I are used, while for $\mathcal{Q}_7^{\leq 11}$ we include the symmetry breaking predicates of type II. Instance $\mathcal{Q}_6^{\leq 9}$ is solved with `CaDiCaL` (1 core), while $\mathcal{Q}_7^{\leq 11}$ using `iCaDiCaL` (128 cores) over the cubes generated by the split algorithm II. Note that the optimal choice of c is dependent on the instance, and while 4 (or symmetrically, $9 - 4 = 5$) works best for $\mathcal{Q}_6^{\leq 9}$, it turns out that 2 (or symmetrically, $11 - 2 = 9$) works best for $\mathcal{Q}_7^{\leq 11}$.

We compare as well the relative impact of the symmetry breaking predicates of type I and type II, showing our results in Table 4. As expected, the type II constraints are the most effective as they divide the search space by $n!$, which is larger than 2^n for $n \geq 4$. Interestingly, our results show a speed-up factor over twice as large as the number of symmetries; while $6! \cdot 2^6 = 46080$, the runtime for $\mathcal{Q}_6^{\leq 9}$ decreases by a factor of $\frac{4679}{0.05} = 93580$.

7.2 Cube and Conquer Experiments

Table 5 presents a comparison of the 3 Cube and Conquer approaches described in Section 4. We observe a linear order speed-up, as our use of Cube and Conquer decreases the runtime by a $\times 50$ factor using 128 cores. Even though `march_cc` performs roughly 4 times faster on

[4]It might be possible to obtain a slight improvement by forcing color 9, as suggested by Table 3 (considering the symmetry between forcing a center color c and $s - c$) we do not have a theoretical explanation for why choosing 9 instead of 2 could be advantageous.

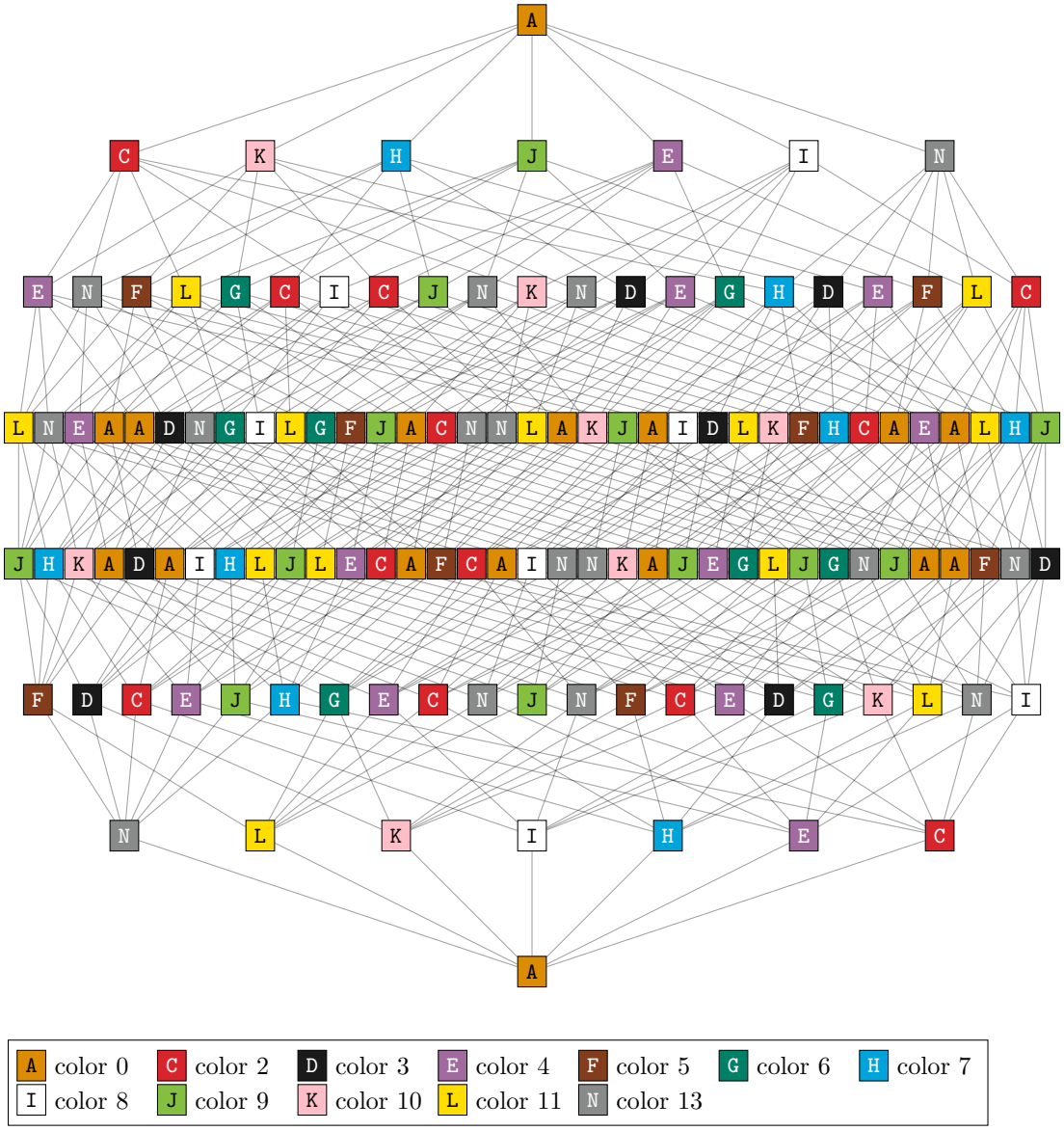

Figure 3: Illustration of a radio 2-coloring for Q_7 with span 13, showing that $\lambda(Q_7) \leq 13$.

Table 4: Runtime in seconds for instances $\mathcal{Q}_6^{\leq 9}$ and $\mathcal{Q}_7^{\leq 10}$ under different kinds of symmetry breaking constraints. All runs were performed with `CaDiCaL` on a single core.

Instance	no SBPs	type I	type II	type I + type II
$\mathcal{Q}_6^{\leq 9}$	4679	102	0.87	0.05
$\mathcal{Q}_7^{\leq 10}$	$>$ 12 hrs	$>$ 12 hrs	183.81	7.64

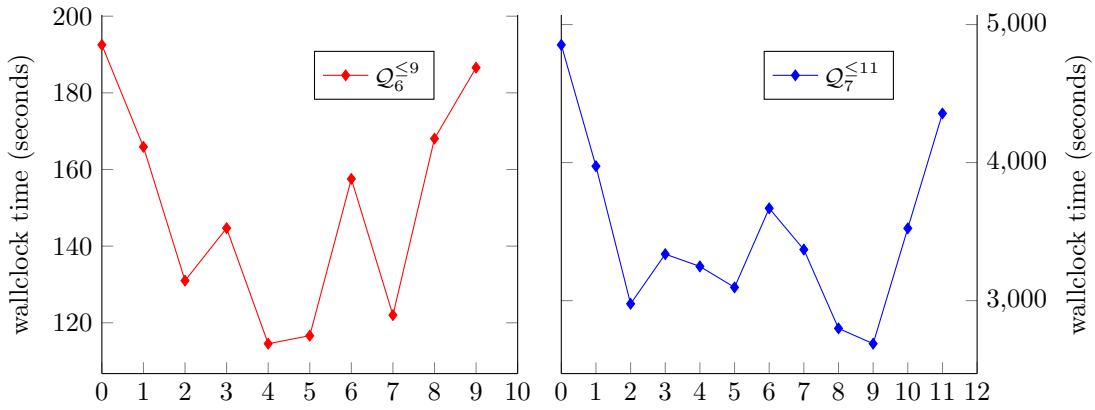

Figure 4: The impact of assigning color c to vertex $\vec{0}$ performance for instances $\mathcal{Q}_6^{\leq 9}$ and $\mathcal{Q}_7^{\leq 11}$. We remark the vertical symmetry in these plots, stemming from the fact that if f is a radio 2-coloring of span s for some graph, then $g(v) := s - f(v)$ also is.

Table 5: Runtime in seconds and (# of cubes) for the instance $\mathcal{Q}_7^{\leq 11}$ under different split algorithms. For the `march_cc` tool we experiment with different values of the *depth* parameter, thus controlling the number of cubes. In all cases, both types of symmetry breaking constraints are applied. All runs were performed with `iCaDiCaL` on 128 cores.

Instance	`march_cc` [16] depth				custom split	no split
	15	16	17	18		
$\mathcal{Q}_7^{\leq 10}$	0.32 (1174)	0.17 (1300)	0.16 (1373)	0.18 (1421)	0.8 (720)	7.64
$\mathcal{Q}_7^{\leq 11}$	> 16 hrs (26826)	> 16 hrs (47762)	> 16 hrs (82426)	> 16 hrs (137729)	2588 (523710)	> 48 hrs

the small instance $\mathcal{Q}_7^{\leq 10}$, our split algorithm is over 12 faster for $\mathcal{Q}_7^{\leq 11}$. In particular, for the instance $\mathcal{Q}_7^{\leq 11}$, at least 5 cubes produced by `march_cc`, at any depth configuration, take over 10 times the entire runtime of our custom split algorithm. This showcases the importance of designing custom split algorithms tailored to the problem at hand.

Note that a free parameter in the custom split algorithm we present is the choice of the set S of vertices over which the cubes are defined. Naturally, the larger the value of $|S|$ is, the more cubes the algorithms will produce. However, as $|S|$ grows, each cube enforces more constraints and thus becomes easier to solve. Consequently, finding the optimal set S is a non-trivial task that we explored experimentally, finding the set $S = \{\texttt{0011}, \texttt{0101}, \texttt{0101}, \texttt{1001}, \texttt{1010}, \texttt{1100}\}$ to perform the best for the instance $\mathcal{Q}_7^{\leq 11}$, while for the smaller instance $\mathcal{Q}_7^{\leq 10}$, the set $S = \{\texttt{0011}, \texttt{0101}, \texttt{0101}\}$, performs better as otherwise the sheer number of cubes becomes a bottleneck.

8 Conclusion and Challenges

We have presented the first study on how SAT solving and automated reasoning techniques can be used for computing optimal radio 2-colorings of hypercube graphs. On the theoretical side, we have proved a gap between $\lambda(Q_n)$ and the upper bound of Whittlesey et al. [10], thus

motivating a more refined theoretical study of $\lambda(Q_n)$; we have shown that the upper bounds of Whittlesey et al. [10] are not tight, and in particular we show this for 7, a number of the form $2^k - 1$, for which the bounds of Theorem 2 and Theorem 3 coincide. This seems to suggest that $2n$ might only be a loose upper bound in general, making the problem of computing $\lim_{n\to\infty} \lambda(Q_n)/n$ an even more exciting line of further research. On the practical side, we have shown the advantages of using symmetry breaking and the Cube and Conquer paradigm. On the one hand, breaking solutions symmetry provides up to a $\times 93\,000$ speed-up factor for UNSAT instances (e.g., $\mathcal{Q}_6^{\leq 9}$, where $n! \cdot 2^n = 46080$), while exploiting internal symmetry provides up to a $\times 100$ speed-up factor for SAT instances. On the other hand, using Cube and Conquer allows for a linear speed-up, as using 128 cores yields over a $\times 50$ speed-up factor. Furthermore, we have shown how a custom split algorithm can be over 50 times faster than the standard tool `march_cc` [16], supporting recent results in other combinatorial problems [17, 18].

Despite our progress, Knuth's Open Problem 1 remains unsolved. Our main goal for future work is thus to solve Open Problem 1, which will probably require new combinatorial insights. At this point, it is unclear whether settling $\lambda(Q_7)$ or $\lambda(Q_{11})$ will be easier. For settling $\lambda(Q_7)$, we conjecture that an extra speed-up factor of around $\times 1000$ on comparable hardware will be needed. For settling $\lambda(Q_{11})$, we believe that cardinality constraints can influence solvers to reason in a similar way to how the lower bound of Jonas [11] was originally proved, which we expect to provide a significant practical speed-up. To conclude, we present a set of challenges for guiding further research on this problem.

Challenge 1. Prove or disprove that $\lambda(Q_n) \leq 2n - 1$ for every $n \geq 4$.

Challenge 2. Prove or disprove that $\lambda(Q_n) \geq n + 4$ for every $n \geq 6$. Note that Jonas proved this for $n = 8$ and $n = 16$ [11]. Observe that this challenge is at least as hard as Knuth's Open Problem 1, because for $n = 11$ it would establish $\lambda(Q_n) = 15$.

Challenge 3. Develop a local search algorithm that can find the currently known upper bounds for $\lambda(Q_n)$ for $n \leq 12$.

Challenge 4. Prove or disprove that, for any n with $\lambda(Q_n) = s$, there exists a mask m, such that at least one radio 2-coloring f of optimal span s exists which satisfies that $f(v \oplus m) = s - f(v)$ for every vertex v.

Challenge 5. Develop a better split algorithm that allows solving instance $\mathcal{Q}_7^{\leq 11}$ in less than 5 minutes on comparable hardware. We believe such a speed-up might be required to solve instance $\mathcal{Q}_7^{\leq 12}$, which we conjecture to be unsatisfiable.

Challenge 6. Extend our result presented in Section 5 by showing that no mapping from Q_7 to Q_k with $k \in \{4, 5, 6\}$ allows for a radio 2-coloring of span 13.

Challenge 7. Develop effective propositional constraints that guide solvers to quickly obtain proofs of $\lambda(Q_n) \geq n + 3$ for $n \leq 15$. Cf. Section 6.

Acknowledgements

The authors thank Donald Knuth for his comments on an earlier version that helped improve the paper. The authors are supported by National Science Foundation grant CCF-2015445.

References

[1] Dimitris Fotakis, Sotiris E. Nikoletseas, Vicky Papadopoulou Lesta, and Paul G. Spirakis. NP-completeness results and efficient approximations for radiocoloring in planar graphs. In *International Symposium on Mathematical Foundations of Computer Science*, 2000.

[2] Pratima Panigrahi. A survey on radio k-colorings of graphs. *AKCE International Journal of Graphs and Combinatorics*, 6(1):161–169, 2009.

[3] W.K. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

[4] Gerard J. Chang and David Kuo. The L(2,1)-labeling problem on graphs. *SIAM Journal on Discrete Mathematics*, 9(2):309–316, 1996.

[5] Tiziana Calamoneri. The L(h, k)-labelling problem: A survey and annotated bibliography. *The Computer Journal*, 49(5):585–608, 2006.

[6] Roger K. Yeh. A survey on labeling graphs with a condition at distance two. *Discrete Mathematics*, 306(12):1217–1231, 2006.

[7] Jerrold R. Griggs and Roger K. Yeh. Labelling graphs with a condition at distance 2. *SIAM Journal on Discrete Mathematics*, 5(4):586–595, 1992.

[8] Jiří Fiala and Jan Kratochvíl. On the computational complexity of the L(2,1)-labeling problem for regular graphs. In Mario Coppo, Elena Lodi, and G. Michele Pinna, editors, *Theoretical Computer Science*, pages 228–236, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[9] Ophir Frieder, Frank Harary, and Peng-Jun Wan. A radio coloring of a hypercube. *International Journal of Computer Mathematics*, 79(6):665–670, 2002.

[10] Marshall A. Whittlesey, John P. Georges, and David W. Mauro. On the λ-number of Q_n and related graphs. *SIAM J. Discret. Math.*, 8:499–506, 1995.

[11] Theodore K. Jonas. *Graph coloring analogues with a condition at distance two: L(2,1) -labellings and list lambda-labellings*. PhD thesis, 1993. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2023-02-20.

[12] E Donald et al. *The Art of Computer Programming. Volume 4, Fascicle 6. Satisfiability*. Addison-Wesley, 2015.

[13] Jiřií Fiala, Ton Kloks, and Jan Kratochvíl. Fixed-parameter complexity of λ-labelings. *Discrete Applied Mathematics*, 113(1):59–72, 2001. Selected Papers: 12th Workshop on Graph-Theoretic Concepts in Computer Science.

[14] Jiří Fiala, Tomáš Gavenčiak, Dušan Knop, Martin Koutecký, and Jan Kratochvíl. Fixed parameter complexity of distance constrained labeling and uniform channel assignment problems. In Thang N. Dinh and My T. Thai, editors, *Computing and Combinatorics*, pages 67–78, Cham, 2016. Springer International Publishing.

[15] A Biere, A Biere, M Heule, H van Maaren, and T Walsh. Handbook of satisfiability: Volume 185 frontiers in artificial intelligence and applications, 2009.

[16] Marijn J. H. Heule, Oliver Kullmann, Siert Wieringa, and Armin Biere. Cube and conquer: Guiding CDCL SAT solvers by lookaheads. In Kerstin Eder, João Lourenço, and Onn Shehory, editors, *Hardware and Software: Verification and Testing*, pages 50–65, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[17] Bernardo Subercaseaux and Marijn J.H. Heule. The packing chromatic number of the infinite square grid is at least 14. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*, volume 236 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[18] Bernardo Subercaseaux and Marijn J. H. Heule. The packing chromatic number of the infinite square grid is 15. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS*

2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part I, volume 13993 of *Lecture Notes in Computer Science*, pages 389–406. Springer, 2023.

[19] Marijn Heule and Toby Walsh. Symmetry within solutions. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 77–82. AAAI Press, 2010.

[20] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, Jul 1948.

[21] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

[22] Armin Biere. Yet another local search solver and Lingeling and friends entering the SAT Competition 2014. In Adrian Balint, Andon Belov, Marijn Heule, and Matti Järvisalo, editors, *Proc. of SAT Competition 2014 – Solver and Benchmark Descriptions*, volume B-2014-2 of *Department of Computer Science Series of Publications B*, pages 39–40. University of Helsinki, 2014.

[23] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing*, Lecture Notes in Computer Science, page 502–518, Berlin, Heidelberg, 2004. Springer.

[24] Shawn T. Brown, Paola Buitrago, Edward Hanna, Sergiu Sanielevici, Robin Scibek, and Nicholas A. Nystrom. *Bridges-2: A Platform for Rapidly-Evolving and Data Intensive Research*, pages 1–4. Association for Computing Machinery, New York, NY, USA, 2021.

A Appendix

A.1 Certificates of $\lambda(Q_7) \leq 13$

We found several certificates of $\lambda(Q_7) \leq 13$ using the parallel local search solver `PalSAT`. Some of the certificates do not use the colors 1 and 12 (an example is shown below). Amongst the certificates avoiding colors 1 and 12 we can observe various internal symmetries. For example, if vertex v is such that $f(v) = c \in \{0, 6, 7, 13\}$ then vertex $f(v \oplus \vec{1}) = c$. Additionally, for all colors c that occur 8 or 16 times (all colors except for 2, 4, 9, and 11) it holds that if vertex $f(v) = c$, then applying the mask `1100110` results in another vertex with color c. For example vertex v_0 and v_{102} both have color 0.

Color 0	0, 7, 25, 30, 42, 45, 51, 52, 75, 76, 82, 85, 97, 102, 120, 127
Color 2	6, 21, 26, 33, 47, 60, 67, 72, 95, 100, 114, 121
Color 3	11, 12, 48, 55, 81, 86, 106, 109
Color 4	2, 20, 31, 37, 46, 57, 71, 73, 90, 96, 115, 124
Color 5	13, 19, 40, 54, 78, 80, 107, 117
Color 6	1, 24, 34, 59, 68, 93, 103, 126
Color 7	15, 22, 44, 53, 74, 83, 105, 112
Color 8	3, 29, 38, 56, 64, 94, 101, 123
Color 9	4, 10, 17, 41, 50, 63, 77, 87, 88, 99, 110, 116
Color 10	27, 28, 32, 39, 65, 70, 122, 125
Color 11	5, 8, 18, 43, 49, 62, 79, 84, 89, 98, 108, 119
Color 13	9, 14, 16, 23, 35, 36, 58, 61, 66, 69, 91, 92, 104, 111, 113, 118

We were not able to find a certificate f such that (a) $\{1, 12\} \cap \text{Range}(f) = \varnothing$, and (b) there exists a mask m for which if vertex $f(v) + f(v \oplus m) = 13$. However, certificates fulfilling condition (b) but not (a) do exist. An example is shown below using mask `0000111`. Moreover, for $c \in \{4, 5, 6, 7, 8, 9\}$, it holds that if $f(v) = c$, then applying mask `0101010` results in another vertex with color c. For example, the vertices v_0 and v_{42} both have color 9.

Color 0	5, 8, 30, 38, 43, 49, 67, 84, 89, 108, 114, 127
Color 1	16, 27, 55, 60, 70, 77, 97, 106
Color 2	1, 12, 22, 34, 47, 57, 75, 85, 88, 100, 115, 126
Color 3	19, 29, 52, 58, 64, 78, 103, 105
Color 4	7, 10, 32, 45, 86, 91, 113, 124
Color 5	9, 21, 35, 63, 79, 80, 101, 122
Color 6	4, 24, 46, 50, 66, 93, 104, 119
Color 7	3, 31, 41, 53, 69, 90, 111, 112
Color 8	14, 18, 36, 56, 72, 87, 98, 125
Color 9	0, 13, 39, 42, 81, 92, 118, 123
Color 10	20, 26, 51, 61, 71, 73, 96, 110
Color 11	6, 11, 17, 37, 40, 62, 76, 82, 95, 99, 116, 121
Color 12	23, 28, 48, 59, 65, 74, 102, 109
Color 13	2, 15, 25, 33, 44, 54, 68, 83, 94, 107, 117, 120

A.2 Certificates of $\lambda(Q_8) \leq 14$

We found several certificates of $\lambda(Q_8) \leq 14$ using the parallel local search solver `PalSAT`. Some of them lack color 7, such as the one below. We did not find any certificates that lacked another color. For some certificates, there exists the following internal symmetry with mask m: $f(v) + f(v \oplus m) = 14$. This works for any mask with 1, 4, and 5 bits set to true. The certificate below has this internal symmetry for the mask `00000001`. This certificate has also another internal symmetry: all colors that are used 16 times (colors 0, 1, 4, 10, 13, and 14) map unto themselves when applying the mask `10111000`. For example, 2 has color 0, so `00000010` $\oplus$ `10111000` = 186 also has color 0.

Color 0	2, 31, 40, 52, 68, 89, 111, 115, 140, 144, 167, 186, 203, 215, 225, 252
Color 1	14, 21, 24, 43, 67, 77, 86, 112, 147, 160, 173, 182, 200, 238, 245, 251
Color 2	9, 18, 36, 49, 62, 64, 91, 103, 106, 125, 133, 138, 159, 163, 184, 198, 209, 220, 233, 242
Color 3	7, 28, 34, 45, 59, 78, 85, 97, 118, 120, 128, 150, 153, 174, 181, 195, 205, 218, 228, 255
Color 4	4, 10, 17, 55, 73, 82, 95, 108, 143, 169, 178, 188, 212, 231, 234, 241
Color 5	13, 22, 33, 46, 56, 71, 92, 98, 117, 123, 131, 149, 154, 164, 191, 192, 206, 217, 237, 246
Color 6	0, 27, 39, 50, 61, 74, 81, 100, 105, 126, 134, 137, 156, 170, 177, 197, 210, 223, 227, 248
Color 8	1, 26, 38, 51, 60, 75, 80, 101, 104, 127, 135, 136, 157, 171, 176, 196, 211, 222, 226, 249
Color 9	12, 23, 32, 47, 57, 70, 93, 99, 116, 122, 130, 148, 155, 165, 190, 193, 207, 216, 236, 247
Color 10	5, 11, 16, 54, 72, 83, 94, 109, 142, 168, 179, 189, 213, 230, 235, 240
Color 11	6, 29, 35, 44, 58, 79, 84, 96, 119, 121, 129, 151, 152, 175, 180, 194, 204, 219, 229, 254
Color 12	8, 19, 37, 48, 63, 65, 90, 102, 107, 124, 132, 139, 158, 162, 185, 199, 208, 221, 232, 243
Color 13	15, 20, 25, 42, 66, 76, 87, 113, 146, 161, 172, 183, 201, 239, 244, 250
Color 14	3, 30, 41, 53, 69, 88, 110, 114, 141, 145, 166, 187, 202, 214, 224, 253

A.3 Proof of Fact 1

Proof of Fact 1. Observe first that for $n \geq 2$ there must be some vertex v such that $f(v) \notin \{0, n+1\}$ in any radio 2-coloring f of Q_n. Then, if we consider $\mathcal{N}(v)$, the set of n neighbors of v, those vertices must receive n different colors, which cannot be in $\{f(v)-1, f(v), f(v)+1\}$. That is, the sets $\{f(v)-1, f(v), f(v)+1\}$ and $\{f(u) \mid u \in \mathcal{N}(v)\}$ are disjoint and their union has size $n+3$. Moreover, given said union contains only non-negative integers, its maximum value m must be at least $n+2$. Said maximum value m must be attained either in $\{f(v)-1, f(v), f(v)+1\}$ or in $\{f(u) \mid u \in \mathcal{N}(v)\}$. In the former case, given that $f(v) \neq n+1$, it must hold that $f(v) \geq n+2$, which is enough to prove the fact. In the latter case, $f(u) \geq n+2$ for some $u \in \mathcal{N}(v)$, which also proves the fact. □