Linearly Ordered Colourings of Hypergraphs

Tamio-Vesa Nakajima
Department of Computer Science, University of Oxford, UK

Stanislav Živný
Department of Computer Science, University of Oxford, UK

Abstract

A linearly ordered (LO) k-colouring of an r-uniform hypergraph assigns an integer from $\{1, \dots, k\}$ to every vertex so that, in every edge, the (multi)set of colours has a unique maximum. Equivalently, for $r = 3$, if two vertices in an edge are assigned the same colour, then the third vertex is assigned a larger colour (as opposed to a different colour, as in classic non-monochromatic colouring). Barto, Battistelli, and Berg [STACS'21] studied LO colourings on 3-uniform hypergraphs in the context of promise constraint satisfaction problems (PCSPs). We show two results.

First, given a 3-uniform hypergraph that admits an LO 2-colouring, one can find in polynomial time an LO k-colouring with $k = O(\sqrt{n \log \log n}/\log n)$, where n is the number of vertices of the input hypergraph. This is established by building on ideas from algorithms designed for approximate graph colourings.

Second, given an r-uniform hypergraph that admits an LO 2-colouring, we establish NP-hardness of finding an LO 3-colouring for every constant uniformity $r \geq 5$. In fact, we determine the precise relationship of polymorphism minions for all uniformities $r \geq 3$, which reveals a key difference between $r = 3, 4$ and $r \geq 5$ and which may be of independent interest. Using the algebraic approach to PCSPs, we actually show a more general result establishing NP-hardness of finding an LO $(k+1)$-colouring for LO k-colourable r-uniform hypergraphs for $k \geq 2$ and $r \geq 5$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Problems, reductions and completeness; Theory of computation → Constraint and logic programming

Keywords and phrases hypegraph colourings, promise constraint satisfaction, PCSP, polymorphisms, minions, algebraic approach

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.128

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Extended version (with stronger results)*: https://arxiv.org/abs/2204.05628

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

1 Introduction

The computational complexity of the *approximate graph colouring* problem [16] is an outstanding open problem in theoretical computer science. Given a 3-colourable graph G on n vertices, is it possible to find a k-colouring of G? On the tractability side, the current best is a polynomial-time algorithm of Kawarabayashi and Thorup [23] that finds a k-colouring with $k = k(n) = n^{0.199}$ colours. On the intractability side, the state-of-the-art for constant k has only recently been improved from $k = 4$, due to Khanna, Linial, and Safra [24] and Guruswami and Khanna [17] to $k = 5$, due to Barto, Bulín, Krokhin, and Opršal [5]. The authors of [5] introduced a general algebraic methodology for studying the computational complexity of so-called promise constraint satisfaction problems (PCSPs). Going beyond the


work in [5], for graphs with a promised higher chromatic number than three, the current best intractability results for constantly many extra colours is due to Wrochna and Živný [27], building on the work of Huang [22].

The situation is much better understood for the approximate *hypergraph* colouring problem with the classic notion of a colouring leaving no edge monochromatic. A celebrated result of Dinur, Regev, and Smyth established that finding an ℓ-colouring of a 3-uniform hypergraph that is k-colourable is NP-hard for every constant $2 \leq k \leq \ell$ [14] (and this also implies the same result on r-uniform hypergraphs for every constant uniformity $r \geq 3$).

Different variants of approximate hypergraph colourings, such as rainbow colourings, were studied, e.g. in [2, 10, 18, 19, 12], but most complexity classifications related to these problems are open. Some intractability results are also known for colourings with a super-constant number of colours. For graphs, conditional hardness was established by Dinur and Shinkar [15]. For hypergraphs, intractability results were obtained by Bhangale [9] and by Austrin, Bhanghale, and Potukuchi [1].

Barto, Battistelli, and Berg have recently studied systematically a certain type of PCSPs on non-Boolean domains and identified a very natural variant of k-colourings of 3-uniform hypergraphs, called *linearly ordered* (LO) k-colourings [4]. A k-colouring of a 3-uniform hypergraph with colours $[k] = \{1, \dots, k\}$ is an LO colouring if, for every edge, it holds that, if two vertices are coloured with the same colour, then the third vertex is coloured with a larger colour. (In the classic non-monochromatic colouring, the requirement is that the third vertex should be coloured with a *different* colour, but not necessarily a larger one.) An LO 2-colouring is thus a "1-in-3" colouring. Barto et al. asked whether finding an LO k-colouring of a 3-uniform hypergraph is NP-hard for a fixed $k \geq 3$ if the input hypergraph is promised to admit an LO 2-colouring.

Contributions

While we do not resolve the question raised in [4], we obtain non-trivial results, both positive (algorithmic) and negative (hardness).

First, we present an efficient algorithm for finding an LO colouring of a 3-uniform hypergraph with super-constantly many colours if an LO 2-colouring is promised to exist. In more detail, for a given 3-uniform hypergraph H on n vertices that admits an LO 2-colouring, we present a polynomial-time algorithm that finds an LO k-colouring of H with $k = k(n) = O(\sqrt{n \log \log n}/\log n)$ colours. As mentioned above, there are only a few results on hypergraph colourings with super-constantly many colours.

Second, we establish intractability of finding an LO 3-colouring of an r-uniform hypergraph if an LO 2-colouring is promised for every constant uniformity $r \geq 5$. In fact, we prove a more general result that finding an LO $(k+1)$-colouring of an r-uniform hypergraph admitting an LO k-colouring is intractable for every constant $k \geq 2$ and $r \geq 5$. This result is based the algebraic approach to PCSPs and in particular on minions [5]. As a matter of fact, we establish the precise relationships of the polymorphism minions of the LO 2- vs 3-colourings on r-uniform hypergraphs for $r \geq 3$, which may be of independent interest. This gives the advertised intractability result but also an impossibility result on certain types of polynomial-time reductions (namely pp-constructions [5]) between LO 2- vs 3-colourings on r-uniform hypergraphs for $r \geq 5$ and $r = 3, 4$.

2 Preliminaries

An *r-uniform hypergraph* H is a pair (V, E) where V is the set of *vertices* of the hypergraph, and $E \subseteq V^r$ is the set of *edges* of the hypergraph. In our context the order of the vertices in each edge is irrelevant. We will allow vertices to appear multiple times in edges; however, we exclude edges of form $(v, \dots, v)$ – such edges would be impossible in the problems we will consider anyway. We say that two distinct vertices u, v are *neighbours* if they both belong to some edge $e \in E$. Let $N(u)$ be the set of neighbours of u. Call a set S an *independent set* of a hypergraph H if and only if no two members of S are neighbours.

A *linearly ordered* (LO) k-colouring of an r-uniform hypergraph $H = (V, E)$ is an assignment $c : V \to [k]$ of colours from $[k] = \{1, \dots, k\}$ to the vertices of H such that, for each edge $(v_1, \dots, v_r) \in E$, the sequence $c(v_1), \dots, c(v_r)$ has a unique maximum. We omit the "k-" if the number of colours is unimportant.

▶ **Example 1.** Consider the hypergraph $H = (V, E)$, where $V = [4] = \{1, 2, 3, 4\}$ and $E = \{(1, 2, 3), (1, 2, 4)\}$. The assignment $c = \{1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 2, 4 \mapsto 2\}$ is an LO 2-colouring, and $c'(x) = x$ is an LO 4-colouring. On the other hand, $c''(x) = 3 - c(x)$ is not an LO colouring at all, since both of the edges have two equal maximal elements when mapped through c''.

Finding an LO k-colouring, for constant $k \geq 3$, of a 3-uniform hypergraph that admits an LO 2-colouring was studied by Barto et al. [4] in the context of promise constraint satisfaction problems (PCSPs), which we define next.

2.1 Promise CSPs

Promise CSPs have been introduced in the works of Austrin, Guruswami, and Håstad [3] and Brakensiek and Guruswami [11]. We follow the notation and terminology of Barto, Bulín, Krokhin, and Opršal [5], adapted to structures consisting of a single relation.

An *r-ary structure* is a pair $\mathbf{D} = (D, R^{\mathbf{D}})$, where $R^{\mathbf{D}} \subseteq D^r$ and D is finite. We call D the *domain* of the structure, and $R^{\mathbf{D}}$ the *relation* of the structure. For two r-ary structures $\mathbf{A}, \mathbf{B}$, a *homomorphism from* $\mathbf{A}$ *to* $\mathbf{B}$ is a function $h : A \to B$ such that, for each $(a_1, \dots, a_r) \in R^{\mathbf{A}}$, we have $(h(a_1), \dots, h(a_r)) \in R^{\mathbf{B}}$. This is written $h : \mathbf{A} \to \mathbf{B}$. If we wish to assert only the existence of such a homomorphism, we write $\mathbf{A} \to \mathbf{B}$.

We now define the search version of the fixed-template PCSP problem. Given two r-ary structures $\mathbf{A} \to \mathbf{B}$, the problem PCSP$(\mathbf{A}, \mathbf{B})$ is the following: given an r-ary structure $\mathbf{I} \to \mathbf{A}$, find a homomorphism $h : \mathbf{I} \to \mathbf{B}$. The decision version of this problem is: given an r-ary structure $\mathbf{I}$, output YES if $\mathbf{I} \to \mathbf{B}$, and output NO if $\mathbf{I} \not\to \mathbf{A}$. Observe that the decision version can be reduced to the search version: to solve the decision version, run an algorithm for the search version, then check if it gives a correct answer. We will use PCSP$(\mathbf{A}, \mathbf{B})$ to mean the decision version when proving hardness, and the search version when showing algorithmic results.

LO colourings can be readily seen as PCSPs. First, observe that an r-uniform hypergraph can be seen as an r-ary structure. Second, define an r-ary structure $\mathbf{LO}_k^r$ with domain $[k]$, and whose relation contains a tuple $(c_1, \dots, c_r)$ if and only if the sequence $c_1, \dots, c_r$ has a unique maximum. Then, an LO k-colouring of an r-uniform hypergraph $\mathbf{H}$ is the same as a homomorphism from $\mathbf{H}$ (viewed as an r-ary structure) to $\mathbf{LO}_k^r$. Thus, the problem of finding an LO k-colouring of an r-uniform hypergraph that has an LO 2-colouring is the same as PCSP$(\mathbf{LO}_2^r, \mathbf{LO}_k^r)$.

In Section 3, we study the computational complexity of $\mathrm{PCSP}(\mathbf{LO}_2^3, \mathbf{LO}_{k(n)}^3)$, were $k(n)$ depends on the input size; here n denotes the number of vertices of the input (3-uniform) hypergraph. As in Example 1, this is obviously possible for $k(n) = n$. As our first contribution, we will present an efficient algorithm with $k(n) = O(\sqrt{n \log\log n}/\log n)$.

In Section 5, we study the computational complexity of $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_{k+1}^r)$ for constant uniformity $r \geq 3$ and constant arity $k \geq 2$. We establish intractability of $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_{k+1}^r)$ for $r \geq 5$ and $k \geq 2$, cf. Corollary 23 (for $k = 2$) and Corollary 25 (for $k \geq 2$) in Section 5. These results are based on the algebraic theory of minions [5], briefly introduced in Section 4. In fact, we establish the precise relationships of the polymorphism minions of $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ for all $r \geq 3$ (cf. Theorem 22 in Section 5).

In the full version of this paper [25] we show stronger results, namely NP-hardness of $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_\ell^r)$ for every constant k and ℓ with $2 \leq k \leq \ell$ and every constant uniformity $r \geq \ell - k + 4$.

3 Algorithmic results

Within this section, we will prove algorithmic results for finding LO colourings of 3-uniform hypergraphs that have LO 2-colourings, using a non-constant number of colours. We will describe two algorithms: a simpler one that uses $O(\sqrt{n})$ colours; and a more complicated one that builds on the first, that uses $O(\sqrt{n \log\log n}/\log n)$ colours. The initial algorithm is inspired by Wigderson's algorithm for the approximate graph colouring problem [26]: like that algorithm it considers a "sparse case" and a "dense case"; however, our algorithm is more complex due to the fact that an LO colouring is different to a normal graph colouring. A particularly salient difference is that our algorithm picks a high-degree *edge*, whereas Wigderson's algorithm picks a high-degree *vertex*. Our second, more complex algorithm refines the first one by selecting more than one edge at a time (similarly to the work of Berger and Rompel [8], although, like Wigderson's algorithm [26], they choose several vertices at a time, not several edges as we do), and by using an improved algorithm for finding independent sets (studied by Halldórsson [21]).

In both algorithms, the general strategy will be to colour some vertices with large colours, then to recursively colour the rest of the hypergraph. The exact choice of "large colour" is made only after the recursive colouring (since only at that point do we know how large the colours need to be). We will not keep track of the bookkeeping needed to do this, in order to make the algorithms easier to explain.

3.1 First algorithm

Fix some 3-uniform hypergraph $H = (V, E)$, with n vertices and m edges. Suppose that this hypergraph admits an LO 2-colouring c^*. We first show a way to extend a partial colouring consistent with c^* until it intersects each edge of H in zero, one or three vertices.

▶ **Lemma 2.** *There exists a polynomial-time algorithm that, given a partial LO 2-colouring c that coincides with c^* on its domain, extends c into a partial LO 2-colouring* extend(c) *that does not intersect any edge of H in exactly 2 vertices and remains consistent with c^* on its domain. The algorithm will also run in polynomial time even if c isn't consistent with c^* or even an LO 2-colouring, in which case it will return an arbitrary extension of c.*

Proof. The following algorithm suffices: while c intersects an edge (x, y, z) in precisely two vertices (say x, y), extend c with $z \mapsto 4 - c(x) - c(y)$. This algorithm clearly extends c until it intersects no edges in precisely 2 vertices. Furthermore, since c^* is an LO 2-colouring, for

each edge (x, y, z) we have $c^*(x) + c^*(y) + c^*(z) = 4$. Thus, if we assume that c is initially consistent with c^*, we can show that each extension keeps c consistent with c^*. We conclude by noting that each extension can be done in polynomial time, and that there are at most n extensions – and that this is the case even if c doesn't coincide with c^* on its domain, or isn't even an LO colouring. ◀

We now show how to find a large independent set if the hypergraph is sparse enough.

▶ **Lemma 3.** *If every edge in H contains a vertex with at most $\Delta \geq 0$ neighbours, then we can find an independent set with $n/(\Delta + 1) = \Omega(n/\Delta)$ vertices in polynomial time. This holds even if Δ happens not to be an integer.*

Proof. This condition implies that there exists at least one vertex with most Δ neighbours: if there exists at least one edge then one of the vertices in it must be this vertex; otherwise, all vertices have $0 \leq \Delta$ neighbours. By adding that vertex to the independent set, removing it and its neighbours from the hypergraph, and repeating this process until no vertices remain, we can find the required independent set. ◀

Our algorithm will colour a certain part of the hypergraph with $O(1)$ colours, colouring the rest recursively, with strictly smaller colours. The following lemma gives us a sufficient goal for the size of what must be coloured in one step.

▶ **Lemma 4.** *A recursive procedure that colours $\Omega(\sqrt{n})$ vertices (where n refers to the* current *size of the hypergraph, not the initial size) with $O(1)$ colours at each step will use $O(\sqrt{n})$ colours overall.*

Proof. Let $T(n)$ be the number of colours needed to colour a hypergraph with at most n vertices. It is sufficient to prove that $T(n) = O(\sqrt{n})$ for n a power of two, since for arbitrary n we can consider the next largest power of two, which is at most $2n$. For some such n, consider how many colours are needed to colour half the hypergraph. At each step until the hypergraph is halved, we colour at least $\Omega(\sqrt{n/2}) = \Omega(\sqrt{n})$ vertices with $O(1)$ colours. Equivalently, we colour half the hypergraph with $O(\sqrt{n})$ colours. Thus we deduce that $T(n) \leq T(n/2) + O(\sqrt{n})$. Applying the Master method of Cormen et al. [13] to the recurrence $U(n) = U(n/2) + \alpha\sqrt{n}$, where α comes from the constant hidden in the recurrence for T, we can deduce that $T(n) = O(\sqrt{n})$. (Applying this method requires satisfying the "regularity condition": $\sqrt{n/2} \leq c\sqrt{n}$ for some $c < 1$ and large enough n; this holds for $c = 1/\sqrt{2}$.) ◀

These results together help us create the algorithm we want.

▶ **Theorem 5.** *There exists a polynomial-time algorithm that finds an LO $O(\sqrt{n})$-colouring for a hypergraph with an LO 2-colouring.*

Proof. We provide a recursive algorithm.

1. If the hypergraph has no edges (x, y, z) where all of x, y and z have at least $\sqrt{n}$ neighbours, then find an independent set with $\Omega(n/\sqrt{n}) = \Omega(\sqrt{n})$ vertices, colour that independent set with a large colour, and recursively colour the rest of the hypergraph with smaller colours. This is possible by Lemma 3.
2. Otherwise, let (x, y, z) be an edge where all of x, y and z have at least $\sqrt{n}$ neighbours.
3. Iterate over $u \in \{x, y, z\}$.
4. Construct a partial colouring c_u where $c_u(u) = 2$ and $c_u(v) = 1$ for $v \in N(u)$.
5. Construct extend(c_u), and check if it is a partial LO 2-colouring. This is possible by Lemma 2.

6. If it is, colour the vertices in the domain of extend(c_u) with two large colours, depending on the colours they got in extend(c_u) (i.e. for some large enough C, colour v with $C + \text{extend}(c_u)(v)$), and then recursively colour the rest of the hypergraph with smaller colours.

We establish the correctness of this algorithm by showing its soundness and completeness.

We begin by showing completeness. The only way we could possibly fail to return something is if, for all $u \in \{x, y, z\}$, the function extend(c_u) is not a proper partial LO 2-colouring. However, for at least one $u \in \{x, y, z\}$, it must be the case that $c^*(u) = 2$, and thus that $c^*(v) = 1$ for $v \in N(u)$. Thus, for this value of u, c^* is consistent with c_u on its domain, and therefore extend(c_u) must be a proper partial LO 2-colouring.

Now we show soundness: whenever we return a colouring, we return an LO $O(\sqrt{n})$-colouring. We first show that we return an LO colouring. Observe that in the case covered in step 1, all the edges that do not intersect the independent set that we find are properly coloured recursively. Furthermore all edges that do intersect the independent set intersect it in exactly one vertex. This vertex is assigned a colour larger than the colours of the other vertices in the edge, so all such edges are also properly coloured. In the case covered in steps 2–6, note that if we return anything, then extend(c_u) is a proper partial LO 2-colouring. Note that all edges intersecting the domain of extend(c_u) in zero or three vertices are correctly coloured (either recursively, or since extend(c_u) is an LO colouring). Furthermore, since the vertices in the domain of extend(c_u) are assigned large colours, those edges that intersect this domain in one vertex are also properly coloured. Thus in this case, since no edge intersects extend(c_u) in two vertices, we also return a proper LO colouring. Thus in all cases we return an LO colouring.

To see that we return an LO $O(\sqrt{n})$-colouring, note that in all cases we colour $\Omega(\sqrt{n})$ vertices with $O(1)$ colours at each iteration (in the first case because we colour an independent set with this size; in the second because c_u must be defined on $N(u)$, which has at least $\sqrt{n}$ vertices). Thus, by Lemma 4, we use $O(\sqrt{n})$ colours overall.

We conclude by noting that this algorithm has recursive depth at most n, and does polynomial work at each step – thus it works in polynomial time. ◀

3.2 Second algorithm

As before, fix some 3-uniform hypergraph $H = (V, E)$, with n vertices and m edges, and suppose that this hypergraph admits an LO 2-colouring c^*. We first reduce our general problem to the special case of finding an LO 3-colouring for a *linear* 3-uniform hypergraph that admits an LO 2-colouring. We call a hypergraph *linear* if no pair of vertices belongs to more than one edge.

▶ **Example 6.** The hypergraph from Example 1 is not linear since vertices 1 and 2 belong to two edges.

The key property of linear hypergraphs we will use is the following: if a vertex v in an r-uniform hypergraph has d neighbours $v_1, \ldots, v_d$, then v is contained in at most d edges. This is the case since each pair (v, v_i) can belong to at most one edge, and $(v, \ldots, v)$ cannot appear as an edge. If H weren't linear, then v could have belonged to at least $\binom{d}{r-1}$ edges, one for each set of $r - 1$ of the d neighbours. (In fact, there are even more possible edges, since vertices are allowed to appear multiple times.)

▶ **Theorem 7.** *If H is a 3-uniform hypergraph with an LO 2-colouring, then we can find, in polynomial time, a linear 3-uniform hypergraph H' with an LO 2-colouring, such that an LO 3-colouring of H can be computed from an LO 3-colouring of H' in polynomial time.*

Proof. The following procedure is sufficient: while H contains two edges of form (x, y, z) and (x, y, z'), identify z with z'. This works because z and z' must be coloured the same in any LO 2-colouring of H (as, in any LO 2-colouring c, $c(x) + c(y) + c(z) = c(x) + c(y) + c(z') = 4$, and thus $c(z) = c(z')$). Thus H' also remains LO 2-colourable. Furthermore, by reversing the identifications, any LO 3-colouring of H' can be made into an LO 3-colouring of H. Finally, this procedure clearly takes polynomial time and results in a linear hypergraph H'. ◀

Thus we will assume that H is linear. We now show how to find a large independent set when the hypergraph is sparse enough, using a series of lemmata and an algorithm of Halldórsson [21].

▶ **Lemma 8.** *Suppose each edge of H has at least one vertex with at most $\Delta \geq 1$ neighbours. Then H has $\Omega(n)$ vertices with at most Δ neighbours.*

Proof. Let x be the number of vertices with at most Δ neighbours. We will show $x = \Omega(n)$ by double counting $|E|$. Note that each edge in $|E|$ contains at least one vertex with at most Δ neighbours; furthermore, each such vertex belongs to at most Δ edges (since H is linear). Thus $|E| \leq x\Delta$. Now note that each vertex with more than Δ neighbours belongs to more than $\Delta/2$ edges, and each edge contains 3 vertices. Thus $3|E| > (n - x)\Delta/2$. Therefore $(n - x)\Delta/6 < x\Delta$, which implies $x = \Omega(n)$. ◀

▶ **Lemma 9.** *Suppose all vertices in H have at most $\Delta \geq 1$ neighbours. Then H has an independent set with at least $\Omega(|E|/\Delta)$ vertices.*

Proof. Let S be the set of values mapped to 2 by c^*; note that S is an independent set. Consider the mapping $f : V \to \mathbb{N}$, where $f(u)$ is zero if $u \notin S$, and otherwise is equal to the number of edges that u belongs to. Since each edge contains exactly one vertex mapped to 2 by c^*, we deduce that $\sum_{u \in V} f(u) = \sum_{u \in S} f(u) = |E|$. Now, each value of $f(u)$ is at most Δ (since H is linear); thus we deduce that $|S|\Delta \geq \sum_{u \in S} f(u) = |E|$, and thus $|S| \geq |E|/\Delta$. ◀

▶ **Theorem 10** ([21, Theorem 4.4])**.** *There exists a polynomial-time algorithm that, if given a graph G with average degree $\overline{d}$, which has an independent set with size s, can find an independent set with size $\Omega(s \log \overline{d}/\overline{d} \log \log \overline{d})$.*

The *primal graph* $P(H)$ of H is a graph with the same vertices as H, and where two vertices are linked by an edge if and only if they are neighbours in H. Since the primal graph preserves neighbours, an independent set in H is independent in $P(H)$ and vice versa.

▶ **Example 11.** The primal graph $P(H)$ of the hypergraph H from Example 1 has $\{1, 2, 3, 4\}$ as vertices and $\{(1, 2), (2, 3), (3, 1), (2, 4), (4, 1)\}$ as edges.

▶ **Theorem 12.** *There exists a polynomial-time algorithm that, if given a linear hypergraph H with an LO 2-colouring where each vertex has most $\Delta = O(\sqrt{n/\log \log n})$ neighbours, can find an independent set with $\Omega(\sqrt{n} \log n/\sqrt{\log \log n})$ vertices.*

Proof. Consider the average degree $\overline{d}$ of the primal graph $P(H)$ of H. Observe that the independent sets of H and $P(H)$ coincide. Suppose $\overline{d} \leq \sqrt[6]{n}$. Thus $P(H)$ has $O(n\sqrt[6]{n})$ edges, and thus $O(n/\sqrt[6]{n}) = o(n)$ vertices of $P(H)$ have degree larger than $\sqrt[3]{n}$. By a simple greedy algorithm applied to the $\Omega(n)$ vertices of $P(H)$ with degree at most $\sqrt[3]{n}$ (repeatedly select the vertex with minimum degree), we can find an independent set of $P(H)$ (and thus of H) with $\Omega(n/\sqrt[3]{n})$ vertices, which is sufficient.

Now suppose $\overline{d} \geq \sqrt[6]{n}$. Note that there exists an independent set with $s = \Omega(|E|/\Delta)$ vertices. Since each edge of H gives rise to at most three edges of $P(H)$, we deduce that $\overline{d} = O(|E|/n)$. Thus $s/\overline{d} = \Omega(|E|/\Delta)/O(|E|/n) = \Omega(n/\Delta) = \Omega(\sqrt{n \log\log n})$. Also, $\log \overline{d}/\log\log \overline{d} \geq \log \sqrt[6]{n}/\log\log \sqrt[6]{n} = \Omega(\log n/\log\log n)$ for large enough n. Thus, the independent set algorithm from [21] applied to $P(H)$ will give us an independent set of $P(H)$ (and thus of H) with $\Omega(\sqrt{n}\log n/\sqrt{\log\log n})$ vertices. ◀

▶ **Corollary 13.** *If we are given instead a hypergraph where each edge has a vertex with at most $\Delta = O(\sqrt{n/\log\log n})$ neighbours, then, by applying the algorithm of Theorem 12 to the subhypergraph formed by taking only the $\Omega(n)$ vertices with at most Δ neighbours, we can still get an independent set with size $\Omega(\sqrt{n}\log n/\sqrt{\log\log n})$.*

As before, our algorithm will be recursive. We now investigate how many vertices must be coloured in one recursive step.

▶ **Lemma 14.** *A recursive procedure that colours $\Omega\left(\sqrt{n}\log n/\sqrt{\log\log n}\right)$ vertices (where n refers to the* current *size of the hypergraph, not the initial size) with $O(1)$ colours at each step will use $O(\sqrt{n\log\log n}/\log n)$ colours overall.*

Proof. Use the same strategy and notation as in Lemma 4. For n a large power of two, until halving n, in one step, we colour $\Omega(\sqrt{n/2}\log(n/2)/\sqrt{\log\log(n/2)}) = \Omega(\sqrt{n}\log n/\sqrt{\log\log n})$ vertices with $O(1)$ colours. The "large enough" part is important to get $n/2$ to the interval where $x \mapsto \sqrt{x}\log x/\sqrt{\log\log x}$ is increasing. Thus, with the same logic as before, $T(n) \leq T(n/2) + O(\sqrt{n\log\log n}/\log n)$ for large enough n. Applying the Master method of Cormen et al. [13] in the same way as in Lemma 4, $T(n) = O(\sqrt{n\log\log n}/\log n)$. (Again, we need the regularity condition: $\sqrt{n/2\log\log(n/2)}/\log(n/2) \leq c\sqrt{n\log\log n}/\log n$ for some $c < 1$ and large enough n; this can be shown for e.g. $c = 0.9$, $n \geq 6$.) ◀

With this result in hand, we now give a stronger algorithm for our problem, inspired by the strategy of Berger and Rompel [8]: we select multiple edges at once, not only one.

▶ **Theorem 15.** *There is a polynomial-time algorithm that finds an LO $O(\sqrt{n\log\log n}/\log n)$-colouring for a hypergraph that has an LO 2-colouring.*

Proof. Consider the following nondeterministic, recursive algorithm.

1. Make H linear by identifying vertices.
2. Let $k = \lceil \log_3 n \rceil$.
3. Let c_0 be an empty partial colouring.
4. For i from 1 to k:
   a. Let (x_i, y_i, z_i) be an edge for which x_i, y_i and z_i do not belong to the domain of c_{i-1}, and the minimum of the numbers of neighbours of x_i, y_i and z_i *not in the domain of* c_{i-1} is as large as possible. If such an edge does not exist, then exit this loop, setting $c_k = c_{i-1}$.
   b. Nondeterministically choose $u_i \in \{x_i, y_i, z_i\}$.
   c. Augment c_{i-1} with $u_i \mapsto 2$ and $v \mapsto 1$ for $v \in N(u_i)$; let the result of "extend"-ing this new function be c_i.

5. If c_k is not a proper LO 2-colouring, then this nondeterministic execution fails.
6. Colour the vertices in the domain of c_k with two large colours according to c_k (i.e. colour u with $C + c_k(u)$ for some large enough C). Remove the domain of c_k from H, letting the result be H'
7. If each edge in H' has a vertex with at most $O(\sqrt{n/\log\log n})$ neighbours, then find an independent set in H' with size $\Omega(\sqrt{n}\log n/\sqrt{\log\log n})$ and colour it with a large colour (but smaller then those used in the previous step), removing it from H'.
8. Recursively colour the rest of the hypergraph with smaller colours.

Since only logarithmically many nondeterministic choices from among a constant number of options are made, and polynomial work is done otherwise, this algorithm can be made into a polynomial-time deterministic one. Thus we show that it is correct, by showing that it is sound and complete.

To show completeness we need to show that at least one sequence of nondeterministic choices doesn't fail. It is therefore sufficient to show that, for one choice of $u_1, \ldots, u_k$, all of c_i are consistent with c^*. We see that, if c_{i-1} is consistent with c^*, then one choice of u_i exists that makes c_i consistent with c^* (the member of $\{x_i, y_i, z_i\}$ that is assigned 2 by c^*). Since c_0 is empty and thus consistent with c^* on its domain, we deduce inductively that one set of nondeterministic choices that keeps c_i consistent with c^* exists, and thus that the algorithm is complete.

Now we show soundness. Just as in the first algorithm, we always colour either by colouring the domain of some LO 2-colouring that doesn't intersect any edges in precisely 2 vertices with two large colours, according to that colouring; or by colouring an independent set with one large colour. (The LO colouring c_k intersects each edge in zero, one or three vertices due to the use of "extend".) We showed in the first algorithm that these procedures, followed by colouring what remains with smaller colours, lead to a proper LO colouring. We now need to show that we use $O(\sqrt{n\log\log n}/\log n)$ colours. To this end, we show that we colour $\Omega(\sqrt{n}\log n/\sqrt{\log\log n})$ vertices in each iteration – since colouring this many vertices with $O(1)$ colours at each recursive step will lead to an $O(\sqrt{n\log\log n}/\log n)$ colouring overall, as shown in Lemma 14. If the condition in step 7 is satisfied, then we clearly colour the required number of vertices; thus suppose it is not satisfied. In this case, there exists an edge e in H' whose vertices have $\Omega(\sqrt{n/\log\log n})$ neighbours. This implies that, during step a., the edge (x_i, y_i, z_i) could have been selected to be e. Since the number of neighbours of a vertex in H' is a lower bound for the number of neighbours of a vertex in H that do not belong to the domain of c_{i-1}, we find that selecting e would have made the minimum of the numbers of neighbours of x_i, y_i and z_i to have been at least $\Omega(\sqrt{n/\log\log n})$. Thus the actual values of x_i, y_i and z_i must all have at least this many neighbours not in the domain of c_{i-1}. Also, an edge is always found in step a. – since e can always be selected. This implies that c_0 is augmented by at least $\Omega(\sqrt{n/\log\log n})$ vertices at each of the $k = \Theta(\log n)$ iterations in step 4; thus c_k is defined on $\Omega(\sqrt{n}\log n/\sqrt{\log\log n})$ vertices. We thus deduce that in this case we also colour enough vertices. We conclude that the algorithm is sound.

Regarding the running time, note that if a (non-deterministic) guess in Step 4.b passes the test in Step 5, then no more guesses are made in this recursive call because the recursive call made in Step 8 cannot fail via our completeness proof. Thus the algorithm runs in polynomial time. ◀

4 Algebraic theory of fixed-template promise CSPs

We recount the algebraic theory of fixed-template PCSPs developed in [5] and specialised to structures with a single relation (of arity r).

The *p-the power* of an r-ary structure $\mathbf{A} = (A, R^{\mathbf{A}})$ is a structure $\mathbf{A}^p = (A^p, R^{\mathbf{A}^p})$ where

$$R^{\mathbf{A}^p} = \{((a_1^1, \dots, a_1^p), \dots, (a_r^1, \dots, a_r^p)) \mid (a_1^1, \dots, a_r^1) \in R^{\mathbf{A}}, \dots, (a_1^p, \dots, a_r^p) \in R^{\mathbf{A}}\}.$$

In other words, a tuple of $R^{\mathbf{A}^p}$ contains r vectors of p elements of A, such that if these are written as a matrix with r rows and p columns, each column is a member of $R^{\mathbf{A}}$. For two r-ary structures $\mathbf{A}, \mathbf{B}$, a p-ary polymorphism from $\mathbf{A}$ to $\mathbf{B}$ is a homomorphism $f : \mathbf{A}^p \to \mathbf{B}$.

▶ **Example 16.** Consider the binary structure $\mathbf{A} = ([2], R^{\mathbf{A}})$, where $R^{\mathbf{A}}$ is the binary disequality relation $\neq$ (restricted to $[2]^2$). The power $\mathbf{A}^5$ has domain $[2]^5$ and relation $\{(\mathbf{a}, \mathbf{b}) \mid \mathbf{a}, \mathbf{b} \in [2]^5, a_i \neq b_i, i = 1, \dots, 5\}$. This relation is constructed as follows: $(\mathbf{a}, \mathbf{b})$ belongs to the relation if and only if every column of a matrix with 5 columns and 2 rows constructed out of $\mathbf{a}, \mathbf{b}$ satisfies $\neq$. The matrix is the following one:

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \\ b_1 & b_2 & b_3 & b_4 & b_5 \end{pmatrix}.$$

Thus, for each column to satisfy $\neq$, we must have $a_i \neq b_i$ for $i = 1, \dots, 5$, as indicated above.

Now, consider a quinary polymorphism $f : \mathbf{A}^5 \to \mathbf{A}$. This is a function $f : [2]^5 \to [2]$ that satisfies the following property: if given a matrix with 2 rows and 5 columns, such that each column is a member of $R^{\mathbf{A}}$, then by applying f to the rows of this matrix we also get a member of $R^{\mathbf{A}}$. For instance, for the matrix

$$\begin{pmatrix} 1 & 2 & 2 & 1 & 1 \\ 2 & 1 & 1 & 2 & 2 \end{pmatrix},$$

we deduce that the pair $(f(1,2,2,1,1), f(2,1,1,2,2)) \in R^{\mathbf{A}}$ i.e. $f(1,2,2,1,1) \neq f(2,1,1,2,2)$. One such polymorphism is given by selecting the values of f from $[2]$ such that $f(x_1, \dots, x_5) \equiv \sum_{i=1}^{5} x_i \pmod 2$.

The real power of this theory comes from *minions.*[1] A *minion* $\mathcal{M}$ is a sequence of sets $\mathcal{M}^{(0)}, \mathcal{M}^{(1)}, \dots$, equipped with an operation that, for $f \in \mathcal{M}^{(p)}$ and $\pi : [p] \to [q]$, yields $f_\pi \in \mathcal{M}^{(q)}$. The operation must satisfy the following conditions:

- For $f \in \mathcal{M}^{(p)}$, if $id : [p] \to [p]$ is the identity on $[p]$, then $f_{id} = f$.
- For $f \in \mathcal{M}^{(p)}$, $\pi : [p] \to [q]$ and $\sigma : [q] \to [t]$, we have $f_{\sigma \circ \pi} = (f_\pi)_\sigma$.

An important class of minion is the *polymorphism minion.* The polymorphism minion $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ for two structures $\mathbf{A}, \mathbf{B}$ with the same arity is a minion where $\mathcal{M}^{(p)}$ is the set of p-ary polymorphisms from $\mathbf{A}$ to $\mathbf{B}$, and where, for $f : A^p \to B, \pi : [p] \to [q]$, f_π is given by $f_\pi(x_1, \dots, x_q) = f(x_{\pi(1)}, \dots, x_{\pi(p)})$. It is not difficult to check that, if $f : \mathbf{A}^p \to \mathbf{B}$ and $\pi : [p] \to [q]$, then $f_\pi : \mathbf{A}^q \to \mathbf{B}$, as required.

In order to be able to relate polymorphism minions with the complexity of PCSPs, we use *minion homomorphisms.*[2] A *minion homomorphism from $\mathcal{M}$ to $\mathcal{N}$* is a mapping ξ that takes each $\mathcal{M}^{(p)}$ to $\mathcal{N}^{(p)}$ and that satisfies the following condition: for any $\pi : [p] \to [q]$ and $f \in \mathcal{M}^{(p)}$, $\xi(f)_\pi = \xi(f_\pi)$. The following theorem links minion homomorphisms to PCSPs. In particular, minion homomorphisms capture precisely a certain type of polynomial-time reductions, know as primitive-positive constructions,[3] studied for CSPs [6] and PCSPs [5].

---

[1] In category-theoretic terms, a minion is a functor from the skeleton of the category of finite sets to the category of sets. The objects of the first category are sets $[p]$ for $p \in \mathbb{N}$, and the arrows are functions between them. The functor equivalent to a minion $\mathcal{M}$ takes $[p]$ to $\mathcal{M}^{(p)}$, and $\pi : [p] \to [q]$ to $f \mapsto f_\pi$.

[2] In category-theoretic terms, a minion homomorphism is just a natural transformation.

[3] Primitive-positive constructions (or pp-constructions, for short) capture so-called "gadget reductions", cf. [6, Section 3].

▶ **Theorem 17** ([5, Theorems 3.1 and 4.12]). *For r-ary structures $\mathbf{A}, \mathbf{B}$ and r'-ary structures $\mathbf{A}', \mathbf{B}'$, a primitive-positive construction-based polynomial-time reduction from* $\mathrm{PCSP}(\mathbf{A}', \mathbf{B}')$ *to* $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ *exists if and only if* $\mathrm{Pol}(\mathbf{A}, \mathbf{B}) \to \mathrm{Pol}(\mathbf{A}', \mathbf{B}')$.

Unfortunately, it is usually a complex task to explicitly construct minion homomorphisms. An auxiliary construction called the *free structure* allows us to construct them more easily. For an arbitrary minion $\mathcal{M}$ and an r-ary structure $\mathbf{A} = (A, R^{\mathbf{A}})$, the *free structure* $\mathbf{F} = \mathbf{F}_{\mathcal{M}}(\mathbf{A})$ is an r-ary structure whose domain is $F = \mathcal{M}^{(|A|)}$. To construct its relation $R^{\mathbf{F}}$, first identify A with $[n]$ for $n = |A|$, and then enumerate the tuples of $R^{\mathbf{A}}$ as vectors $\mathbf{r}^1, \ldots, \mathbf{r}^k$, where $k = |R^{\mathbf{A}}|$. Construct functions $\pi_1, \ldots, \pi_r : [k] \to [n]$ where $\pi_i(j) = \mathbf{r}^j_i$. (If we were to arrange $\mathbf{r}^1, \ldots, \mathbf{r}^k$ into a matrix with r rows and k columns, then $\pi_i(1), \ldots, \pi_i(k)$ is the i-th row of the matrix.) Now, the tuple $(f_1, \ldots, f_r)$, where $f_1, \ldots, f_r \in \mathcal{M}^{(n)}$, belongs to $R^{\mathbf{F}}$ if and only if, for some $f \in \mathcal{M}^{(k)}$ we have $f_i = f_{\pi_i}$.

▶ **Example 18.** Consider some polymorphism minion $\mathcal{M}$ and the ternary structure $\mathbf{LO}_3^3$. We will construct $\mathbf{F} = \mathbf{F}_{\mathcal{M}}(\mathbf{LO}_3^3)$. The domain is $\mathcal{M}^{(3)}$. To construct the relation of $\mathbf{F}$, we first arrange the 15 tuples of $R^{\mathbf{LO}_3^3}$ into a matrix with 3 rows and 15 columns:

$$\begin{pmatrix} 2 & 1 & 1 & 3 & 1 & 1 & 3 & 2 & 3 & 1 & 2 & 1 & 3 & 2 & 2 \\ 1 & 2 & 1 & 1 & 3 & 1 & 2 & 3 & 1 & 3 & 1 & 2 & 2 & 3 & 2 \\ 1 & 1 & 2 & 1 & 1 & 3 & 1 & 1 & 2 & 2 & 3 & 3 & 2 & 2 & 3 \end{pmatrix}.$$

Row i of this matrix can be seen as a function $\pi_i : [15] \to [3]$. Now the relation $R^{\mathbf{F}}$ contains precisely the tuples $(f_{\pi_1}, f_{\pi_2}, f_{\pi_3})$ for all $f \in \mathcal{M}^{(15)}$. Substituting the definition for f_{π_i}, we find that these polymorphisms $f_{\pi_1}, f_{\pi_2}, f_{\pi_3}$ are:

$$(x, y, z) \mapsto f\,(y, x, x, z, x, x, z, y, z, x, y, x, z, y, y)$$
$$(x, y, z) \mapsto f\,(x, y, x, x, z, x, y, z, x, z, x, y, y, z, y)$$
$$(x, y, z) \mapsto f\,(x, x, y, x, x, z, x, x, y, y, z, z, y, y, z)$$

Observe that the matrix and the arguments of f are actually arranged in the same configuration, with 1 replaced by x, 2 by y and 3 by z.

The following theorem connects minion homomorphisms and the free structure.

▶ **Theorem 19** ([5, Lemma 4.4]). *If $\mathcal{M}$ is a minion and $\mathbf{A}, \mathbf{B}$ are r-ary structures, the homomorphisms $h : \mathbf{F}_{\mathcal{M}}(\mathbf{A}) \to \mathbf{B}$ are in a (natural) 1-to-1 correspondence to the minion homomorphisms $\xi : \mathcal{M} \to \mathrm{Pol}(\mathbf{A}, \mathbf{B})$.*[4] *As a consequence, $\mathbf{F}_{\mathcal{M}}(\mathbf{A}) \to \mathbf{B}$ if and only if $\mathcal{M} \to \mathrm{Pol}(\mathbf{A}, \mathbf{B})$.*

5 Hardness results

In this section we will investigate the hardness of $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_{k+1}^r)$. First, we will establish that $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_{k+1}^r)$ is NP-hard for $r \geq 5$ and $k \geq 2$. In particular, this proves intractability of $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ for $r \geq 5$. Second, we will show that $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ with $r \geq 5$ *cannot* be reduced to $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ with $r = 3$ and $r = 4$ using primitive-positive constructions (i.e. gadget reductions [5]).

[4] In category-theoretic terms, $\mathbf{F}_{-}(\mathbf{A})$ and $\mathrm{Pol}(\mathbf{A}, -)$ are functors between (in opposite directions) the category of r-ary structures and the category of minions, and $\mathbf{F}_{-}(\mathbf{A}) \dashv \mathrm{Pol}(\mathbf{A}, -)$.

We will use a hardness result of Guruswami and Trevisan [20] (stated as Theorem 20 below) on the *1-in-r exact hitting set*. An instance of this problem is an r-uniform hypergraph, and an admissible solution is a subset of its vertices. We are then asked to maximise the number of edges e for which exactly one vertex of e belongs to the chosen subset. An instance is called *α-satisfiable* if it is possible to find a subset of vertices such that a proportion of α of all the edges satisfy this condition. An instance is called *satisfiable* if it is 1-satisfiable. We will call the problem of distinguishing a satisfiable 1-in-r exact hitting set instance from one that is not even α-satisfiable the *α-gap 1-in-r exact hitting set problem*. Guruswami and Trevisan state a weaker version of the following theorem, but their proofs establish the following.

▶ **Theorem 20** ([20, Theorem 10, Theorem 12, Lemma 13])**.** *For any $\epsilon > 0$, for large enough r, the $(1/(e-\epsilon))$-gap 1-in-r exact hitting set problem is* NP*-hard.*

▶ **Corollary 21.** $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ *is* NP*-hard for large enough r.*

Proof. Observe that the instances of (1/2)-gap 1-in-r exact hitting set and $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ are both r-uniform hypergraphs. We show that the identity function is a reduction from the first problem to the second. For completeness, note that a satisfiable instance of 1-in-r exact hitting set is also immediately a YES-instance of $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$. For soundness, consider a YES-instance of $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ i.e. an r-uniform hypergraph with an LO 2-colouring and thus also an LO 3-colouring. Under this colouring each edge either has a unique vertex assigned 3, or a unique vertex assigned 2; thus, at least half of the edges must contain exactly one 2, or at least half of the edges must contain exactly one 3. We deduce that taking either the set of vertices assigned 2, or the set of vertices assigned 3, gives us a solution that satisfies at least half the edges of the hypergraph, viewed as a hitting set instance. ◀

How can we now leverage this basic hardness result to other values of r? We will use chains of minion homomorphisms to do this. We define $\mathcal{M}_r = \mathrm{Pol}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$. Our main result in this section is the following.

▶ **Theorem 22.** *The relationships between the minions $\mathcal{M}_r = \mathrm{Pol}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ for $r \geq 3$ are as shown in Figure 1, i.e., all homomorphisms not drawn or implied do not exist.*

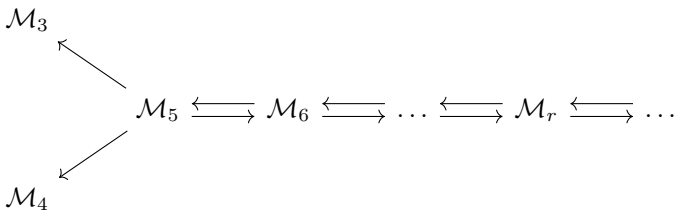

Figure 1 Minion homomorphism order of minions $\mathcal{M}_r$ for $r \geq 3$.

Combining Theorems 17 and 22 gives the following.

▶ **Corollary 23.** $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ *is* NP*-hard for $r \geq 5$. Moreover, there is no polynomial-time reduction using pp-constructions from $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ to $\mathrm{PCSP}(\mathbf{LO}_2^3, \mathbf{LO}_3^3)$ and from $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ to $\mathrm{PCSP}(\mathbf{LO}_2^4, \mathbf{LO}_3^4)$ for $r \geq 5$.*

A simple proof shows the following:

▶ **Theorem 24.** *For every $r \geq 5$ and $\ell > k \geq 2$, $\mathrm{Pol}(\mathbf{LO}_{k+1}^r, \mathbf{LO}_{\ell+1}^r) \to \mathrm{Pol}(\mathbf{LO}_k^r, \mathbf{LO}_\ell^r)$.*

Proof. Consider any p-ary polymorphism $f \in \mathrm{Pol}(\mathbf{LO}^r_{k+1}, \mathbf{LO}^r_{\ell+1})^{(p)}$. Consider the value of f for inputs $a_1, \ldots, a_p \in [k]$; due to the following matrix with $r \geq 3$ rows

$$\begin{pmatrix} a_1+1 & \ldots & a_p+1 \\ a_1 & \ldots & a_p \\ \vdots & \ddots & \vdots \\ a_1 & \ldots & a_p \end{pmatrix},$$

we can deduce that $f(a_1, \ldots, a_p) < f(a_1+1, \ldots, a_p+1) \in [\ell+1]$. This implies that $f(a_1, \ldots, a_p) \in [\ell]$. We claim this implies that f, restricted to $[k]^p$, is a polymorphism of $\mathrm{Pol}(\mathbf{LO}^r_k, \mathbf{LO}^r_\ell)$. Consider matrix of inputs a^j_i where $i \in [p]$, $j \in [r]$, such that each column $a^1_i, \ldots, a^r_i$ is a tuple of $\mathbf{LO}^r_k$. Thus each column is also a tuple of $\mathbf{LO}^r_{k+1}$. Since f is a polymorphism of $\mathrm{PCSP}(\mathbf{LO}^r_{k+1}, \mathbf{LO}^r_{\ell+1})$, we deduce that

$$(f(a^1_1, \ldots, a^1_p), \ldots, f(a^r_1, \ldots, f^r_p))$$

is a tuple of $\mathbf{LO}^r_{\ell+1}$ i.e. has a unique maximum. But we already know these values belong to $[\ell]$. Since they have a unique maximum, they are a tuple of $\mathbf{LO}^r_\ell$. Thus f, restricted to $[k]^p$, is a polymorphism of $\mathrm{Pol}(\mathbf{LO}^r_k, \mathbf{LO}^r_\ell)$.

We now claim that the map $f \mapsto f|_{[k]^p}$ taking a p-ary polymorphism to its restriction on $[k]^p$ is a minion homomorphism $\mathrm{Pol}(\mathbf{LO}^r_{k+1}, \mathbf{LO}^r_{\ell+1}) \to \mathrm{Pol}(\mathbf{LO}^r_k, \mathbf{LO}^r_\ell)$. To see why, consider any polymorphism $f \in \mathrm{Pol}(\mathbf{LO}^r_{k+1}, \mathbf{LO}^r_{\ell+1})^{(p)}$ and a function $\pi : [p] \to [q]$. What we need to prove is that

$$(f_\pi)|_{[k]^p} = (f|_{[k]^p})_\pi.$$

But note that, for $x_1, \ldots, x_p \in [k]$,

$$\begin{aligned}((f_\pi)|_{[k]^p})(x_1, \ldots, x_p) = f_\pi(x_1, \ldots, x_p) &= f(x_{\pi(1)}, \ldots, x_{\pi(p)}) \\ &= (f|_{[k]^p})(x_{\pi(1)}, \ldots, x_{\pi(p)}) = (f|_{[k]^p})_\pi(x_1, \ldots, x_p).\end{aligned}$$

This concludes the proof. ◀

Theorem 24 and Corollary 23 imply the following:

▶ **Corollary 25.** $\mathrm{PCSP}(\mathbf{LO}^r_k, \mathbf{LO}^r_{k+1})$ *is* NP*-hard for* $r \geq 5$ *and* $k \geq 2$.

In order to construct the minion homomorphisms, we first exhibit a simple necessary and sufficient condition for the existence of a minion homomorphism to $\mathrm{Pol}(\mathbf{LO}^r_2, \mathbf{LO}^r_k)$, and a sufficient condition for such a homomorphism to not exist.

▶ **Lemma 26.** *Fix* $r \geq 3$ *and* $k \geq 3$. *Consider any polymorphism minion* $\mathcal{M}$. *For any element* $f \in \mathcal{M}^{(r)}$, *let* $f_1(x, y) = f(y, x, \ldots, x)$, $f_2(x, y) = f(x, y, x, \ldots, x)$, $\ldots$, $f_r(x, y) = f(x, \ldots, x, y)$. *Now,* $\mathcal{M} \to \mathrm{Pol}(\mathbf{LO}^r_2, \mathbf{LO}^r_k)$ *if and only if there exists some* $\omega : \mathcal{M}^{(2)} \to [k]$ *such that, for all* $f \in \mathcal{M}^{(r)}$, *there exists a unique maximum value among* $\omega(f_1), \ldots, \omega(f_r)$.

Proof. We construct $\mathbf{F}_{\mathcal{M}}(\mathbf{LO}^r_2)$. The tuples of the relation of $\mathbf{LO}^r_2$ are all the r-dimensional vectors containing exactly one 2, with the other entries equal to 1. We can arrange these tuples into an r-by-r matrix where the diagonal contains 2 and all the other elements are 1. Replacing 1 with x and 2 with y, and applying f, we get the definitions of $f_1, \ldots, f_r$. Thus the relation of $\mathbf{F}_{\mathcal{M}}(\mathbf{LO}^r_2)$ contains precisely the tuples of form $(f_1, \ldots, f_r)$ for $f \in \mathcal{M}^{(r)}$.

Thus our condition amounts to the existence of a homomorphism $\omega : \mathbf{F}_{\mathcal{M}}(\mathbf{LO}^r_2) \to \mathbf{LO}^r_k$. By Theorem 19, this is equivalent to $\mathcal{M} \to \mathrm{Pol}(\mathbf{LO}^r_2, \mathbf{LO}^r_k)$. ◀

For the next lemma, call a function f *block-symmetric* with respect to a partition of the arguments of f into blocks if f is not changed by arbitrarily permuting the arguments in the blocks. For instance $(x, y, z) \mapsto x + y$ is block-symmetric with respect to the blocks $\{x, y\}$ and $\{z\}$.

▶ **Lemma 27.** *Fix $r \geq 3$ and a polymorphism minion $\mathcal{M}$. If $\mathcal{M}$ has a block-symmetric polymorphism of arity r with respect to a partition in which all blocks consist of at least two elements, then $\mathcal{M} \not\to \mathcal{M}_r$.*

Proof. We use the same notation as in the proof of Lemma 26. Let f be this block-symmetric polymorphism, and note that no polymorphism among $f_1, \ldots, f_r$ appears exactly once, due to block-symmetry. Thus, for any $\omega : \mathcal{M}^{(2)} \to [3]$, the sequence $\omega(f_1), \ldots, \omega(f_r)$ does not have a unique maximum. Applying the previous lemma gives us the required result. ◀

▶ **Theorem 28.** *For any $r \geq 5$, $\mathcal{M}_r \to \mathcal{M}_{r+1}$.*

Proof. We will establish the statement via Lemma 26. We will use the assignment $\omega : \mathcal{M}^{(2)} \to [3]$ given by $\omega(f) = f(1, 2)$. To check that this satisfies our condition, we need to investigate $\mathcal{M}_r^{(r+1)}$. Observe that an $(r + 1)$-ary polymorphism $f \in \mathcal{M}_r^{(r+1)}$ is a function from $[2]^{r+1}$ to $[3]$; if it is applied to the rows of an $r \times (r+1)$ matrix whose columns are tuples of the relation of $\mathbf{LO}_2^r$ (i.e. they contain one 2 and otherwise are 1), then the resulting values contain a unique maximum. Similarly to Barto et. al. [4], we view f as a function from the powerset of $[r + 1]$ to $[3]$. (Thus, the input tuple $(1, 2, 1, 2)$ is seen as equivalent to the input set $\{2, 4\}$.) Under this view, f is a polymorphism if and only if, for any partition $A_1, \ldots, A_r$ of $[r + 1]$, the sequence $f(A_1), \ldots, f(A_r)$ has a unique maximum element. (Observe that each part A_i corresponds to a row in the matrix mentioned earlier). To show that ω satisfies our condition, what we need to prove is that the sequence $f\{1\}, \ldots, f\{r + 1\}$ has a unique maximum. We have three cases depending on the maximum of these values.

Maximum is 3. Suppose that at least one of $f\{1\}, \ldots, f\{r+1\}$ is 3. Without loss of generality, say $f\{1\} = 3$. Suppose for contradiction that another of the values is also 3; without loss of generality, say $f\{2\} = 3$. But consider the partition $\{1\}, \{2\}, \{3, \ldots, r + 1\}, \emptyset, \ldots, \emptyset$ of $[r + 1]$ into r sets. Since f is a polymorphism, the images of these sets must have a unique maximum value. But if $f\{1\} = f\{2\} = 3$ this is impossible! So if $f\{1\} = 3$, then this is the unique maximum of the sequence $f\{1\}, \ldots, f\{r + 1\}$.

Maximum is 2. Now, suppose that all of $f\{1\}, \ldots, f\{r + 1\}$ are either 1 or 2, and that at least one of these (say $f\{1\}$) is 2. Suppose for contradiction that another value (say $f\{2\}$) is also 2. Due to the partitions $\{1\}$, $\{2\}$, $\{3, 4\}$, $\{5\}$, $\ldots$, $\{r + 1\}$; $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5, 6\}$, $\{7\}$, $\ldots$, $\{r + 1\}$ we deduce that $f\{3, 4\} = f\{5, 6\} = 3$. This is impossible because of the partition $\{1, 2, 7, \ldots, r + 1\}, \{3, 4\}, \{5, 6\}, \emptyset, \ldots, \emptyset$. Thus in this case $f\{1\} = 2$ is the unique maximum of the sequence $f\{1\}, \ldots, f\{r + 1\}$.

Maximum is 1. Finally, suppose that $f\{1\} = \ldots = f\{r + 1\} = 1$. Consider the partitions $\{1, 2\}$, $\{3\}$, $\ldots$, $\{r + 1\}$; $\{1\}$, $\{2\}$, $\{3, 4\}$, $\{5\}$, $\ldots$, $\{r + 1\}$; and $\{1\}$, $\ldots$, $\{4\}$, $\{5, 6\}$, $\{7\}$, $\ldots$, $\{r + 1\}$. Since all the singletons have image 1, the two-element sets here must have image 2 or 3. At least two of them must therefore have the same image; if that image is 3, then we have a contradiction like in the last case. Thus suppose, without loss of generality, that $f\{1, 2\} = f\{3, 4\} = 2$. Considering the partition $\{1, 2\}, \{3, 4\}, \{5\}, \ldots, \{r + 1\}, \emptyset$, we find that $f(\emptyset) = 3$. But this cannot happen, due to partition $\emptyset, \ldots, \emptyset, \{1, \ldots, r + 1\}$. Thus this case is impossible.

Our assignment of values to polymorphisms of $\mathcal{M}_r^{(2)}$ is correct. Therefore we deduce that $\mathcal{M}_r \to \mathcal{M}_{r+1}$. ◀

▶ **Theorem 29.** *For any $r \geq 3, k \geq 3$,* $\mathrm{Pol}(\mathbf{LO}_2^{r+2}, \mathbf{LO}_k^{r+2}) \to \mathrm{Pol}(\mathbf{LO}_2^r, \mathbf{LO}_k^r)$. *In particular, for $k = 3$,* $\mathcal{M}_{r+2} \to \mathcal{M}_r$.

Proof. We use the same notation convention as in Theorem 28, and we take $\omega(f) = f(1, 2)$. Thus, we want to prove that, if f is a function that takes subsets of $[r]$ to $[k]$ such that, for any partition $A_1, \ldots, A_{r+2}$, the sequence $f(A_1), \ldots, f(A_{r+2})$ has a unique maximum, then the sequence $f\{1\}, \ldots, f\{r\}$ has a unique maximum. But consider the partition $\{1\}, \ldots, \{r\}, \emptyset, \emptyset$, and note that the largest value cannot be $f(\emptyset)$ (since $f(\emptyset)$ appears twice). Thus we deduce that one of $f\{1\}, \ldots, f\{r\}$ is the maximum, and furthermore that this maximum is strictly larger than all the other values in this sequence. Thus, $\mathcal{M}_{r+2} \to \mathcal{M}$. ◀

Proof of Theorem 22. We have $\mathcal{M}_{r+1} \to \mathcal{M}_{r+2} \to \mathcal{M}_r$, so $\mathcal{M}_{r+1} \to \mathcal{M}_r$ for every $r \geq 5$ by Theorems 28 and 29. Thus, $\mathcal{M}_5 \rightleftarrows \mathcal{M}_6 \rightleftarrows \ldots$; furthermore, by Theorem 29, we have $\mathcal{M}_5 \to \mathcal{M}_3$. Finally, by Theorem 28 and Theorem 29, we have $\mathcal{M}_5 \to \mathcal{M}_6 \to \mathcal{M}_4$.

It remains to show that (i) $\mathcal{M}_3 \not\to \mathcal{M}_r$ for any $r \geq 4$ and that (ii) $\mathcal{M}_4 \not\to \mathcal{M}_r$ for any $r \geq 3$, $r \neq 4$. An auxiliary function will be useful for both: let $f : \mathbb{N} \to \mathbb{N}$ map 0 to 2, 1 to 1, and all other values to 3.

Regarding (i), note that $\mathcal{M}_3$ has a quaternary block-symmetric polymorphism with respect to a partition in which all blocks have size 2, viz. $(a, b, c, d) \mapsto f(a + b)$. Thus, by Lemma 27, $\mathcal{M}_3 \not\to \mathcal{M}_4$. Since $\mathcal{M}_r \to \mathcal{M}_5 \to \mathcal{M}_4$ for $r \geq 4$, we deduce that $\mathcal{M}_3 \not\to \mathcal{M}_r$.

Regarding (ii), note that $\mathcal{M}_4$ has a ternary symmetric polymorphism, viz. $(a, b, c) \mapsto f(a + b + c)$. Thus, by Lemma 27, $\mathcal{M}_4 \not\to \mathcal{M}_3$. Since $\mathcal{M}_r \to \mathcal{M}_5 \to \mathcal{M}_3$ for $r \geq 3, r \neq 4$, we deduce that $\mathcal{M}_4 \not\to \mathcal{M}_r$. ◀

In Theorem 28, using Theorem 17, we showed that hardness of $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ for some large r implies hardness of $\mathrm{PCSP}(\mathbf{LO}_2^5, \mathbf{LO}_3^5)$. We now show that the same is true for $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_k^r)$: hardness of $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_k^r)$ for some large r implies hardness of $\mathrm{PCSP}(\mathbf{LO}_2^{r(k)}, \mathbf{LO}_k^{r(k)})$, for some $r(k)$ that depends only on k.

▶ **Theorem 30.** *For $k \geq 4, r \geq 4k - 3$,* $\mathrm{Pol}(\mathbf{LO}_2^r, \mathbf{LO}_k^r) \to \mathrm{Pol}(\mathbf{LO}_2^{r+1}, \mathbf{LO}_k^{r+1})$.

Note that the lower bound on r in the statement is $4k - 3$, which is worse than the bound for $k = 3$ in Theorem 28. We do not know whether a smaller bound is possible for $k > 3$.

Proof. Fix some k and let $\mathcal{N}_r = \mathrm{Pol}(\mathbf{LO}_2^r, \mathbf{LO}_k^r)$. We show that $\mathcal{N}_r \to \mathcal{N}_{r+1}$ for $r \geq 4k - 3$.

Use the same notation as in Theorem 28, and the same choice of ω; what we want to show is that if $f \in \mathcal{N}_r^{(r+1)}$ then the sequence $f\{1\}, f\{2\}, \ldots, f\{r + 1\}$ has a unique maximum. Recall that f satisfies the property that, if $A_1, \ldots, A_r$ is a partition of $\{1, \ldots, r + 1\}$, then $f(A_1), \ldots, f(A_r)$ has a unique maximum. We now split into three cases, depending on the maximum of $f\{1\}, \ldots, f\{r + 1\}$.

Maximum is k. The maximum must be unique in this case. If we suppose, contrarily and without loss of generality, that $f\{1\} = f\{2\} = k$, then the partition $\{1\}, \{2\}, \{3, \ldots, r + 1\}, \emptyset, \ldots, \emptyset$ yields our contradiction.

Maximum is 1. This case is impossible. Consider the partitions $\{1, 2\}, \{3\}, \ldots, \{r + 1\}$; $\{1\}, \{2\}, \{3, 4\}, \{5\}, \ldots, \{r + 1\}$; $\ldots$; $\{1\}, \ldots, \{4k - 3, 4k - 2\}, \{4k - 1\}, \ldots, \{r + 1\}$. These partitions exist since $4k - 2 \leq r + 1$. All of the two-element sets must be mapped to some values in $\{2, \ldots, k\}$ (they cannot be 1 since then all the parts in the previous partitions map to 1), and there are $2k - 1$ two-element sets. Thus by pidgeonhole principle three sets must be mapped to the same value. Thus suppose $f\{1, 2\} = f\{3, 4\} = f\{5, 6\}$. Considering the partition $\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7\}, \ldots, \{r + 1\}, \emptyset, \emptyset$ we find that none of the images through f can be the unique maximum (since the first three and the last two are equal, and all the rest are 1). Thus we have the required contradiction.

Maximum is neither. Suppose that the maximum is $1 < k' < k$, and suppose that the maximum is not unique. Thus without loss of generality let $f\{1\} = f\{2\} = k'$. Now consider the partitions $\{1\},\{2\},\{3,4\},\{5\},\dots,\{r+1\};\dots;\{1\},\{2\},\dots,\{4k-4\},\{4k-3,4k-2\},\{4k-1\},\dots,\{r+1\}$. The two-element sets must be mapped to a value from $\{k'+1,\dots,k\}$ i.e. to one of at most $k-2$ values, and there are $2k-2$ sets. Thus at least three of these sets are assigned the same value. Suppose without loss of generality that $f\{3,4\} = f\{5,6\} = f\{7,8\}$. Then the partition $\{1\},\{2\},\{3,4\},\{5,6\},\{7,8\},\{9\},\dots,\{r+1\},\emptyset,\emptyset$ gives us a contradiction: the images of the first two sets, the next three sets, and the last two sets form equal blocks; and all other sets have images that are not greater than the images of the first two sets.

Thus, since our choice of ω is valid, by Lemma 26, we find that $\mathcal{N}_r \to \mathcal{N}_{r+1}$. ◀

6 Conclusions

The question about the complexity of $\mathrm{PCSP}(\mathbf{LO}_2^3, \mathbf{LO}_k^3)$ for constant $k \geq 3$ raised in [4] stays open. The complexity of $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_\ell^r)$ is open except for the hardness obtained in our work. In this paper, we show NP-hardness for $k \geq 2$, $\ell = k+1$, and any $r \geq 5$. In the full version of this paper [25], we show NP-hardness for $2 \leq k \leq \ell$ and any $r \geq \ell - k + 4$.

The minion homomorphisms (and lack thereof) between the polymorphism minions $\mathrm{Pol}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ for various values of r have interesting implications for the complexity of PCSPs more broadly. First, if one were to prove that this problem is hard for $r = 3$ or $r = 4$, then our results imply that hardness in linearly ordered colourings does not necessarily follow from minion homomorphisms and thus in particular cannot be obtained via "gadget reductions" [5]. This is in contrast to the case of (non-promise) CSPs, where it is known [7] (cf. also [6])[5] that all NP hardness can be shown using minion homomorphisms.[6] Second, our results show that, if one proves that $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_\ell^r)$ is hard for some large arity r, then it is hard $\mathrm{PCSP}(\mathbf{LO}_2^{r'}, \mathbf{LO}_\ell^{r'})$ for some arity $r' = r'(\ell)$ that depends only on ℓ.

Going beyond the realm of fixed-template PCSPs [5] (which limits the number of colours by a constant), what is the smallest function $k(n)$ for which $\mathrm{PCSP}(\mathbf{LO}_2^3, \mathbf{LO}_{k(n)}^3)$ is solvable efficiently? There is no clear reason to believe that positive result from the present paper with $k(n) = O(\sqrt{n \log\log n}/\log n)$ is optimal.

References

1 Per Austrin, Amey Bhangale, and Aditya Potukuchi. Simplified inpproximability of hypergraph coloring via t-agreeing families, 2019.

2 Per Austrin, Amey Bhangale, and Aditya Potukuchi. Improved inapproximability of rainbow coloring. In *Proc. 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 1479–1495, 2020. doi:10.1137/1.9781611975994.90.

3 Per Austrin, Venkatesan Guruswami, and Johan Håstad. $(2+\epsilon)$-Sat is NP-hard. *SIAM J. Comput.*, 46(5):1554–1573, 2017. doi:10.1137/15M1006507.

4 Libor Barto, Diego Battistelli, and Kevin M. Berg. Symmetric Promise Constraint Satisfaction Problems: Beyond the Boolean Case. In *Proc. 38th International Symposium on Theoretical Aspects of Computer Science (STACS'21)*, volume 187 of *LIPIcs*, pages 10:1–10:16, 2021. doi:10.4230/LIPIcs.STACS.2021.10.

[5] [6] uses the terminology of height 1 identities.

[6] This would not be the first time though in the context of PCSPs. A recent example of the same phenomenon comes from [27] for the problem of approximate graph colouring.

5 Libor Barto, Jakub Bulín, Andrei A. Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. *J. ACM*, 68(4):28:1–28:66, 2021. doi:10.1145/3457606.

6 Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and how to use them. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.1.

7 Libor Barto, Jakub Opršal, and Michael Pinsker. The wonderland of reflections. *Isr. J. Math*, 223(1):363–398, February 2018. doi:10.1007/s11856-017-1621-9.

8 Bonnie Berger and John Rompel. A better performance guarantee for approximate graph coloring. *Algorithmica*, 5(3):459–466, 1990. doi:10.1007/BF01840398.

9 Amey Bhangale. NP-Hardness of Coloring 2-Colorable Hypergraph with Poly-Logarithmically Many Colors. In *Proc. 45th International Colloquium on Automata, Languages, and Programming (ICALP'18)*, volume 107 of *LIPIcs*, pages 15:1–15:11, 2018. doi:10.4230/LIPIcs.ICALP.2018.15.

10 Joshua Brakensiek and Venkatesan Guruswami. New hardness results for graph and hypergraph colorings. In *Proc. 31st Conference on Computational Complexity (CCC'16)*, volume 50 of *LIPIcs*, pages 14:1–14:27, 2016. doi:10.4230/LIPIcs.CCC.2016.14.

11 Joshua Brakensiek and Venkatesan Guruswami. Promise Constraint Satisfaction: Algebraic Structure and a Symmetric Boolean Dichotomy. *SIAM J. Comput.*, 50(6):1663–1700, 2021. doi:10.1137/19M128212X.

12 Joshua Brakensiek and Venkatesan Guruswami. The quest for strong inapproximability results with perfect completeness. *ACM Trans. Algorithms*, 17(3):27:1–27:35, 2021. doi:10.1145/3459668.

13 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

14 Irit Dinur, Oded Regev, and Clifford Smyth. The hardness of 3-uniform hypergraph coloring. *Comb.*, 25(5):519–535, September 2005. doi:10.1007/s00493-005-0032-4.

15 Irit Dinur and Igor Shinkar. On the Conditional Hardness of Coloring a 4-Colorable Graph with Super-Constant Number of Colors. In *Proc. 13th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'10)*, volume 6302 of *LNCS*, pages 138–151. Springer, 2010. doi:10.1007/978-3-642-15369-3_11.

16 M. R. Garey and D. S. Johnson. The complexity of near-optimal graph coloring. *J. ACM*, 23(1):43–49, 1976. doi:10.1145/321921.321926.

17 Venkatesan Guruswami and Sanjeev Khanna. On the hardness of 4-coloring a 3-colorable graph. *SIAM J. Discret. Math*, 18(1):30–40, 2004. doi:10.1137/S0895480100376794.

18 Venkatesan Guruswami and Euiwoong Lee. Strong inapproximability results on balanced rainbow-colorable hypergraphs. *Comb.*, 38(3):547–599, 2018. doi:10.1007/s00493-016-3383-0.

19 Venkatesan Guruswami and Sai Sandeep. Rainbow coloring hardness via low sensitivity polymorphisms. *SIAM J. Discret. Math*, 34(1):520–537, 2020. doi:10.1137/19M127731X.

20 Venkatesan Guruswami and Luca Trevisan. The complexity of making unique choices: Approximating 1-in- k SAT. In *Proc. 8th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'05)*, volume 3624 of *LNCS*, pages 99–110. Springer, 2005. doi:10.1007/11538462_9.

21 Magnús M. Halldórsson. Approximations of weighted independent set and hereditary subset problems. *J. Graph Algorithms Apppl.*, 4(1):1–16, 2000. doi:10.7155/jgaa.00020.

22 Sangxia Huang. Improved hardness of approximating chromatic number. In *Proc. 16th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'13)*, pages 233–243. Springer, 2013. doi:10.1007/978-3-642-40328-6{_}17.

23 Ken-ichi Kawarabayashi and Mikkel Thorup. Coloring 3-colorable graphs with less than $n^{1/5}$ colors. *J. ACM*, 64(1):4:1–4:23, 2017. doi:10.1145/3001582.

24 Sanjeev Khanna, Nathan Linial, and Shmuel Safra. On the hardness of approximating the chromatic number. *Comb.*, 20(3):393–415, March 2000. doi:10.1007/s004930070013.

25 Tamio-Vesa Nakajima and Stanislav Živný. Linearly ordered colourings of hypergraphs, 2022. arXiv:2204.05628.

26 Avi Wigderson. Improving the performance guarantee for approximate graph coloring. *J. ACM*, 30(4):729–735, 1983. doi:10.1145/2157.2158.

27 Marcin Wrochna and Stanislav Živný. Improved hardness for H-colourings of G-colourable graphs. In *Proc. 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 1426–1435, 2020. doi:10.1137/1.9781611975994.86.