THEORY OF COMPUTING
www.theoryofcomputing.org

A logarithmic approximation of linearly ordered colourings

Johan Håstad Björn Martinsson Tamio-Vesa Nakajima
Stanislav Živný

February 18, 2026

Abstract: A linearly ordered (LO) k-colouring of a hypergraph assigns to each vertex a colour from the set $\{0, 1, \ldots, k-1\}$ in such a way that each hyperedge has a unique maximum element. Barto, Batistelli, and Berg conjectured that it is NP-hard to find an LO k-colouring of an LO 2-colourable 3-uniform hypergraph for any constant $k \geq 2$ (STACS'21) but even the case $k = 3$ is still open. Nakajima and Živný gave polynomial-time algorithms for finding, given an LO 2-colourable 3-uniform hypergraph, an LO colouring with $O^*(\sqrt{n})$ colours (ICALP'22) and an LO colouring with $O^*(\sqrt[3]{n})$ colours (ToCT 2023). Very recently, Louis, Newman, and Ray gave an SDP-based algorithm with $O^*(\sqrt[5]{n})$ colours (FSTTCS'24). We present two simple polynomial-time algorithms that find an LO colouring with $O(\log_2(n))$ colours, which is an exponential improvement.

A conference version of this paper appeared in the Proceedings of the 27th International Conference on Approximation Algorithms for Combinatorial Optimization Problems. This work was supported by UKRI EP/X024431/1, by a Clarendon Fund Scholarship and by the Knut and Alice Wallenberg Foundation. For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. All data is provided in full in the results section of this paper.

ACM Classification: F.2.2, G.2.1

AMS Classification: 68W25, 68Q25, 68W20

Key words and phrases: Linear ordered colouring, Hypergraph, Approximation, Promise Constraint Satisfaction Problems

1 Introduction

Given a graph G, the *graph k-colouring* problem asks to find a colouring of the vertices of G by colours from the set $\{0,1,\dots,k-1\}$ in such a way that no edge is monochromatic. The *approximate graph colouring problem* asks, given a k-colourable graph G, to find an ℓ-colouring of G, where $\ell \geq k$. For $k=3$, the state-of-the-art results are NP-hardness of the case $\ell=5$ [3] and a polynomial-time algorithm for finding a colouring with $\ell = O(n^{0.19747})$ colours, where n is the number of vertices of the input graph G [15]. For non-monochromatic colourings of hypergraphs, it is known that finding an ℓ-colouring of a k-colourable r-uniform hypergraph is NP-hard for any constant $\ell \geq k \geq 2$ and $r \geq 3$ [11], and also some positive results are known for colourings with super-constantly many colours, e.g. [17, 16, 9].

A new promise hypergraph colouring problem was identified in [2]. Given a 3-uniform hypergraph H, a colouring of the vertices of H with colours from the set $\{0,1,\dots,k-1\}$ is called a *linearly ordered* (LO) k-colouring if every edge e of H satisfies the following: if two vertices of e have the same colour then the third colour is larger. More generally, a colouring of a hypergraph H is an LO colouring if every edge of H has a unique maximum colour. (Note that the two definitions coincide for 3-uniform hypergraphs.) Barto et al. conjectured that finding an LO ℓ-colouring of a 3-uniform hypergraph that admits an LO k-colouring is NP-hard for every constant $\ell \geq k \geq 2$ [2] but even the case $k=2$ and $\ell=3$ is open. Nakajima and Živný established NP-hardness for some regimes of the parameters k,ℓ,r [19, 20] (where r is the uniformity of the input hypergraph) and, very recently, Filakovký et al. [13] showed NP-hardness of the case $k=3$, $\ell=4$, $r=3$. More importantly for this paper, Nakajima and Živný also considered finding an LO $f(n)$-colouring of an LO 2-colourable 3-uniform hypergraph with n vertices and presented polynomial-time algorithms with $f(n) = O(\sqrt{n\log\log n}/\log n)$ [19] and $f(n) = O(\sqrt[3]{n\log\log n/\log n})$ [20]. Very recently, Louis, Newman, and Ray [18] have given a polynomial-time SDP-based algorithm with $f(n) = O^*(\sqrt[5]{n})$ colours.

As our main result, we improve their results by an exponential factor.

Theorem 1.1. *There is an algorithm which, if given a 3-uniform hypergraph H with $n \geq 4$ vertices and m edges that admits an LO 2-colouring, finds an LO $\log_2(n)$-colouring of H in time $O(n^3+nm)$.*

In fact we present two different algorithms that return colourings using $O(\log n)$ colours. Both are based on solving the natural system of linear equations implied by the existence of an LO 2-colouring. In one case, the system is solved modulo 2, and in the other case, the system is solved over the rational numbers.

While the H which we are given as input is 3-uniform, we will need the notion we define next in greater generality; hence we define it for general hypergraphs. For each edge $\{x_1,\dots,x_r\}$ of H, we write an equation $v_{x_1}+\dots+v_{x_r}=1$ where we initially use equality modulo 2 but as stated above we later use the same system over the rational numbers. Let A be this set of equations, written as a matrix with m rows and n columns. (Note that A is the *incidence matrix* of H.) Thus v is a solution if and only if $Av = 1^m$. Clearly a valid LO 2-colouring gives one solution but in the general case, the system has a large dimensional affine space as its set of solutions and the desired solution is hard to find.

This is the journal version of the conference paper [14]. The main change is that we have a new method for using the solution over the rational numbers. This improvement reduces the number of colours used by the algorithm in Section 3 from $2\log_2 n$ to $1.5\log_2 n$ colours.

Related work. While the notion of LO colouring was first identified in the context of promise problems in [2], it is identical to the notion of *unique-maximum colouring* [7] of a hypergraph, and similar to that of *conflict-free colouring*, introduced by Even et al. [12] and Smorodinsky [21]. (In a conflict-free colouring, every hyperedge must contain a unique value, but this value need not be the maximum). There is also the related notion of a *graph unique-maximum colouring*, also known as *ordered colouring* or *vertex ranking* [8], which is the same as a unique-maximum colouring of the path hypergraph of a graph (i.e. the hypergraph whose edges are paths of the graph). We refer the reader to [23] for more on conflict-free and unique-maximum colouring.

It is worth mentioning Smorodinsky's framework for unique-maximum colouring of a hypergraph (see [22] where it is proposed for conflict-free colouring, and also [23] where it is extended to unique-maximum colouring), which does the following: Given a hypergraph H, find a non-monochromatic colouring with few colours, select the largest cardinality colour class, colour it in our unique-maximum colouring with the minimum colour, then continue recursively. Smorodinsky uses this algorithm to find unique-maximum colourings of graphs where it is *hereditarily* easy to find non-monochromatic colourings with few colours — in particular graphs that come from geometric situations. Unfortunately in our case this does not happen. Consider for example the hypergraph H with vertices v_i for $i \in [k]$ and w_{ij} for $i, j \in [k]$; and edges (v_i, v_j, w_{ij}). Suppose we apply Smorodinsky's algorithm to this hypergraph, and at the first step we find the non-monochromatic colouring given by $v_i \mapsto 0$ and $w_{ij} \mapsto 1$. Then at the next step Smorodinsky's algorithm must colour the clique on $v_1, \ldots, v_k$ — this basically means that it outputs a colouring with $\Theta(\sqrt{n})$ colours, since the number of vertices is $\Theta(k^2)$.[1] Thus even if we ignore the fact that finding the colouring on the graph that we get after the first step is NP-hard in general (note that we could encode finding a colouring of any graph we wanted rather than just the clique), we do not necessarily get a small number of colours by this framework. The algorithm for conflict-free colourings presented in [12] is similar. In each step, it chooses a set of vertices that intersects each edge e either (i) in one vertex, or (ii) in $< |e|$ vertices. (These two cases are disjoint only when $|e| = 1$.) It then colours these vertices with one colour, throwing away all edges coloured by exactly one vertex. To generalise this to unique-maximum colourings one would need to keep all edges except those for which exactly one vertex remains — thus this algorithm does not work in our setting for the same reason.

While these algorithms have some similarity with ours, we critically do not find a non-monochromatic colouring of our hypergraph H at each step. Indeed, what we do at any particular step either immediately solves an edge or leaves it completely intact — this lets us keep the property of LO 2-colourability for what is left to colour intact.

Let us return to the world of promise problems. Our problem is a *promise CSP* [3] of the form "given a 3-uniform LO 2-colourable hypergraph, find a homomorphism to a fixed 3-uniform hypergraph H". In [2], the computational complexity of this problem was classified for all 3-vertex hypergraphs H except for the H that encodes LO 3-colouring. This gap in their findings is what motivated the authors to introduce the linearly ordered colouring problem. In [10], the authors classified the problem for any H whose edges do not contain 3 distinct elements; i.e. all edges are of the form (x, x, y). In particular, this

[1] Observe that even the linearisation trick from [20] does nothing for this hypergraph, as it is already linear — i.e. every two edges intersect in at most one vertex. The essence of the trick is to identify vertices x and y if there exist vertices a, b and edges (x, a, b) and (y, a, b), as such vertices must have the same colour in any LO 2-colouring.

implies that the *rainbow-free*[2] LO 2- vs. LO k-colouring problem is NP-hard for every fixed k. However, note that the case of $k = 3$ was already shown to be NP-hard by [2].

2 Algorithm based on equations modulo 2

In this section all linear equations are taken modulo 2. For the following, given a set S of positive integers, an S-uniform hypergraph is a hypergraph where all edges have sizes taken from S. We first prove the following subprocedure of the main algorithm.

Lemma 2.1. *There is an algorithm which, if given a $\{2,3\}$-uniform hypergraph H with n vertices and m edges that admits an LO 2-colouring and such that the implied linear system of equations $Av = 1^m$ does not fix the value of any variable, outputs a subset T of vertices that intersects edges of size three in zero or two vertices and edges of size two in exactly one vertex. Moreover, we have $|T| \geq n/2$. The algorithm runs in $O(n^3 + nm)$ time.*

Proof. We first describe a randomised version of our algorithm, and then derandomise it. The set of solutions to $Av = 1^m$ is an affine space and hence a generic solution can be written as $v = v^0 + \sum_{i=1}^{r} a_i v^i$ for a basic solution v^0, linearly independent solutions to the homogeneous system v^i, and field elements (in this case bits) a_i. The fact that no variable is fixed implies that for each vertex x there is some positive i such that $v_x^i = 1$.

For the randomised algorithm choose $a_1, \ldots, a_r$ to be independent identically distributed uniformly random bits, and set T to be the set of variables x, such that $v_x = 0$. Clearly T satisfies the conclusion of the lemma as in each edge we have an odd number of ones. For every vertex x, there exists positive i such that $v_x^i = 1$ — hence, due to the influence of $a_i v_x^i$, we see that x is included in T with probability $\frac{1}{2}$. Thus, on average T contains half the vertices.

Now, we derandomise this algorithm using the method of conditional expectations. Go through the variables a_i in increasing order and fix its value once and for all. Fixing the value of a_i determines the value(s) of some v_x while other values remain undetermined. For each value being determined $v_x^i = 1$ and hence one value of a_i gives the final value 0 and the other gives final value 1. Set a_i such that at least half the determined values are 0. After we have fixed all a_i this way, we have a final solution with at least $n/2$ zeroes.

The bottleneck of the running time of this algorithm is solving the linear system of equations. This can be done in the advertised running time since every equation has $O(1)$ entries. □

Proof of Theorem 1.1. As a preliminary step, we eliminate any variable determined by the system $Av = 1^m$. Note that if the colour of a vertex is determined by the system $Av = 1^m$, then this vertex must have that same colour in *all* LO 2-colourings. Fix these variables once and for all and eliminate them from the equation system. For all vertices that have been given the colour 1, we set the colours of the two other vertices in all of its edges to be 0. This process of identifying fixed variables and eliminating them is then repeated until the system $Av = 1^m$ contains no variables fixed to a constant. At any fixed point of this process, for every edge, either all vertices in that edge are fixed (and the edge has a unique-maximum as required), or exactly one vertex in it is fixed to 0.

[2] In the *rainbow-free* variant, the goal is to find a colouring without rainbow edges, i.e. no edge can contain 3 distinct colours.

Now, remove all coloured vertices from the hypergraph H, shrinking the edges they belonged to. The remaining hypergraph will no longer be 3-uniform (it will be $\{2,3\}$-uniform though), but importantly it will still be LO 2-colourable. Our goal is still to LO colour the remaining hypergraph, since any edge partially coloured by the preliminary step above must have had exactly one vertex v fixed to 0; and hence, if we LO colour the edge that resulted from removing v, this leads to an LO colouring of the original hypergraph when v is assigned 0.

Consider the following algorithm, where i starts at 0.

1. If the hypergraph H has at most, say, 20 vertices, find an LO colouring of H by brute force using colours i and $i+1$. (It exists since H is LO 2-colourable.)

2. Otherwise, find the subset T guaranteed by Lemma 2.1.

3. Colour the vertices in T by colour i. Remove the vertices in T from H. Remove all edges that intersect T from H. Increment i by 1.

4. Repeat.

Note that $|T| \geq n/2$ and thus within $-4+\log_2 n$ repetitions we reach the first case. Each step adds one colour and we get two additional colours from the final brute-force colouring for a total of at most $\log_2 n$ colours. The output is correct as the first time some vertex in an edge is coloured, for edges with three vertices exactly one more vertex in the same edge is coloured at the same time, and for edges with two vertices only that vertex is coloured at that time. The remaining vertex is given a higher colour and hence the edge is correctly coloured.

For the time complexity, we again note that it is dominated by the time needed to solve the linear system of equations. Since n decreases by a factor of 2 at every step, and the cost of the inner loop is $O(n^3+nm)$, this gives us the required time complexity — even ignoring the fact that m also is decreased. □

Note that the number 20 selected above can be increased to any number that is $O(\log n)$ and the algorithm remains polynomial time (since we must compute the colouring for a subgraph of this size by brute force). If we stop the algorithm at B vertices, then we save $\log B + \Theta(1)$ colours, since this is how many colours the algorithm would have used to colour the last B vertices. By setting $B = \Theta(\log n)$, we can thus save $\Theta(1) + \log\log n$ colours while keeping run time of the algorithm polynomial in n.

A slight variant can be obtained by instead counting the number of remaining edges with no coloured vertex. Once we have no more such vertices, we colour all remaining vertices with the next colour. For such edge, a random solution v gives the four sets of values $(0,0,1)$, $(0,1,0)$, $(1,0,0)$ and $(1,1,1)$ with equal probabilities. Thus the number of edges decreases, on average, by a factor 4 for each iteration. (Note that all the edges of size 2 are solved in the first iteration, so there is no need to count them.) It is easy to achieve this deterministically by conditional expectations. Once we have not edge with three uncoloured vertices we can colour all remaining uncoloured vertices with the next colour. We state the conclusion as a theorem.

Theorem 2.2. *There is an algorithm which, if given a 3-uniform hypergraph H with n vertices and $m \geq 1$ edges that admits an LO 2-colouring, finds an LO $(2+\frac{1}{2}\log_2(m))$-colouring of H in time $O(n^3+nm)$.*

Remark 2.3. Our algorithm has some similarity with algorithms for *temporal* CSPs [4]. Note that a rainbow-free LO ω-colouring (which means an LO colouring, but with no restriction on the number of colours; also, *rainbow tuples* i.e. tuples (x,y,z) with $x \neq y \neq z \neq x$ are disallowed) is a temporal CSP; to solve it, one finds a subset that could be the smallest colour (by solving mod-2 equations as above), sets that colour, then continues recursively. The difference is that for a rainbow-free LO ω-colouring one does not care about the number of colours, so one can find any nonempty set of vertices to set the lowest colour to, whereas in our problem we are trying to find a large set of this kind. We note that the algorithm of [20] also uses this approach when setting "small colours".

Remark 2.4. We remark that the subprocedure of our algorithm computes the exclusive or of two vectors of bits. Thus the algorithm runs very fast in practice — on most architectures hundreds of operations of this kind are done at one time by (i) packing the bits within a larger word and (ii) using SIMD instructions.

3 Algorithm using $\mathbb{Q}$

In this section we present a more complicated algorithm which uses more colours. This might seem pointless, and indeed it might be. On the other hand the ideas used are slightly different and hence there might be situations where the ideas of this section can turn out to be useful. It is also curious to see that we can use the same system of linear equations, now over the rational numbers, in a rather different way. The algorithm here is in fact essentially saying that we can always use the unbalanced case of [18]. As this eliminates many complications and in particular makes it possible to completely avoid any semi-definite programming, we state all facts needed in the current section rather than refer to the very similar statements in [18]. As already stated, all arithmetic in this section is over the rational numbers. In this situation, no variables can be determined as we can set v^0 to have all coordinates equal to $1/3$.

We study the homogeneous system $Av = 0^m$ and by the assumption of LO 2-colourability it has a solution, w, with coordinates either $-\frac{1}{3}$ or $\frac{2}{3}$. Let us first show how solutions over the rational numbers can be used to find LO colourings. This is the same lemma used in the unbalanced case of [18].

Lemma 3.1. *Suppose we have a solution, u, to the homogeneous system where M is the maximal value of the absolute value of a coordinate and $m > 0$ is the minimal absolute value. Then, we can LO colour with $2 + \log_2(M/m)$ colours.*

Proof. For notational convenience let us instead require that the minimal colour in each edge should be unique. We can simply reverse the order of the colours at the end. By scaling we can assume $M = 1$. We use even colours for positive coordinates and we give x the colour 2ℓ if v_x is at most $2^{-(2\ell-1)}$ and strictly larger than $2^{-(2\ell+1)}$. For negative coordinates we use $2\ell+1$ as the colour if v_x is between $-2^{-2\ell}$ (inclusive) and $-2^{-(2\ell+2)}$ (non-inclusive). Let us verify that this gives a correct colouring.

Take an edge (x,y,z) and suppose both x and y get the same colour 2ℓ. Then by the linear equation of the edge $v_z < -2^{-2\ell}$ and thus z has a colour below 2ℓ. The case of two vertices of odd colour is similar and as the bound on the number of colours is immediate, the lemma follows. □

To find a solution which to apply Lemma 3.1 we first find a set of solutions $\{v^i\}_{i=1}^n$ to homogeneous system $Av = 0^m$. We require that the ith coordinate of v^i, i.e. v^i_i, equals $1/2$ and the maximum absolute value of any coordinate of v^i is at most 1. As $-3w/2$ or $3w/4$ satisfies these conditions such solution exists and some solution can be found by linear programming. Define $u = \sum_{i=1}^{n} y_i v^i$ where y_i are independent uniform variables in $[-1,1]$.

Lemma 3.2. *The vector u has the following two properties:*

1. $\Pr[\min_i |u_i| \leq \frac{1}{4n}] \leq \frac{1}{2}$

2. $\Pr[\max_i |u_i| \geq 2\sqrt{n \ln n}] \leq \frac{2}{n}$

Proof. We prove the two bounds separately.

1. We first show that $\Pr[|u_i| \leq \frac{1}{4n}] \leq \frac{1}{2n}$. A union bound, summing over all $i = 1, \ldots, n$, then implies that $\Pr[\min_i |u_i| \leq \frac{1}{4n}] \leq \frac{1}{2}$.

   Observe that $u_i = \sum_j y_j v_i^j$, where $v_i^i = \frac{1}{2}$, $|v_i^j| \leq 1$ and y_j is distributed uniformly at random in $[-1,1]$. Suppose we sample y_i last, and that the sum $\sum_{j \neq i} y_j v_i^j$ has evaluated to α. Conditional on this fact, u_i is uniformly distributed in $[\alpha - \frac{1}{2}, \alpha + \frac{1}{2}]$. The probability that $|u_i| \leq \frac{1}{4n}$ is then given by the length of the intersection of the interval $[\alpha - \frac{1}{2}, \alpha + \frac{1}{2}]$ with the interval $[-\frac{1}{4n}, \frac{1}{4n}]$ — which is at most $\frac{1}{2n}$, as required.

2. This follows from standard Chernoff bounds as each coordinate of u is the sum of n independent random variables with mean 0 and absolute value at most 1. The probability that such a variable takes the value at least t is at most $\exp(-t^2/2n)$. For a proof of this well known fact see appendix A of [1]. Applying this with $t = 2\sqrt{n \ln n}$ and using the union bound proves this part of the lemma. (The factor of 2 comes from the fact that we need to bound the *absolute value* of u_i, not just u_i itself.) □

Assuming that $n \geq 8$, with probability at least $\frac{1}{4}$, we can apply Lemma 3.1 with $M = 2\sqrt{n \ln n}$ and $m = \frac{1}{4n}$ and we conclude.

Theorem 3.3. *Using the system of linear equations over the rational numbers we can find, with probability at least* $\frac{1}{4}$ *and in polynomial time, an LO colouring with* $1.5 \log_2 n + O(\log \log n)$ *colours.*

This algorithm is less efficient compared to the algorithm of the previous section. The main computational cost is solving linear programs which is more complicated than solving linear systems of equations. Our bound for the number of colours is also worse. It might be possible to find a random solution to the linear system in a different way but if an average coordinate has value $\Theta(1)$ then, heuristically, it seems likely that some of the n coordinates would be at distance $O(1/n)$ from zero resulting in ratio $\Theta(n)$, between the highest and lowest absolute value. Thus it seems unlikely that the algorithm based on the rational numbers would beat the algorithm described in the previous section. We suspect that it is possible to derandomise also this algorithm by conditional expectations, but as this would be complicated let us ignore this possibility.

Nevertheless, we include this algorithm using $\mathbb{Q}$ because it is, in some sense, complementary to the first one. While the first one colours "bottom-up" (i.e. always sets the minimal colour again and again), the algorithm using $\mathbb{Q}$ colours "top-down" (i.e. sets the maximal colour again and again). It is interesting that the algorithm from [20] combined these two approaches, whereas these algorithms stick to only one approach each and get exponentially better results. The improvement seems to come from using the power of random solutions to the linear system A, whether solved over $\mathbb{Z}_2$ or $\mathbb{Q}$.

As a final observation in this section let us note that defining a colouring by the sign of the vector u we get a standard (non-monochromatic) 2-colouring of the hypergraph. This gives an alternative algorithm to that of [6, 5].

4 Concluding remarks

Our algorithms indicate that LO 2-colouring is quite different from many other colouring problems. The key property that we use in our algorithm is that the constraint implies a linear constraint. The analysis of the algorithms also heavily relies on the fact that we study 3-uniform hypergraphs.

It is tempting to think that the proposed methods would extend to other constraint satisfaction problems where we are guaranteed that a solution must satisfy a linear constraint. We have so far been unable to find an interesting such example.

Acknowledgement. This paper is a merger of independent work by Håstad and Martinsson, and by Nakajima and Živný respectively. We are grateful to Venkat Guruswami for noting and informing the authors of the fact that we independently had found the same algorithm. We thank Dömötör Pálvölgyi for informing us that LO colourings have been studied under the name of *unique-maximum* colourings.

References

[1] NOGA ALON AND JOEL H. SPENCER: *The Probabilistic Method.* Wiley, New York, 2004. 7

[2] 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*, pp. 10:1–10:16, 2021. [doi:10.4230/LIPIcs.STACS.2021.10, arXiv:2010.04623] 2, 3, 4

[3] 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, arXiv:1811.00970] 2, 3

[4] MANUEL BODIRSKY AND JAN KÁRA: The complexity of temporal constraint satisfaction problems. *J. ACM*, 57(2):9:1–9:41, 2010. [doi:10.1145/1667053.1667058] 6

[5] JOSHUA BRAKENSIEK AND VENKATESAN GURUSWAMI: An algorithmic blend of LPs and ring equations for promise CSPs. In *Proc. 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*, pp. 436–455, 2019. [doi:10.1137/1.9781611975482.28, arXiv:1807.05194] 8

[6] 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, arXiv:1704.01937] 8

[7] PANAGIOTIS CHEILARIS, BALÁZS KESZEGH, AND DÖMÖTÖR PÁLVÖLGYI: Unique-maximum and conflict-free coloring for hypergraphs and tree graphs. *SIAM J. Discret. Math.*, 27(4):1775–1787, 2013. [doi:10.1137/120880471] 3

[8] PANAGIOTIS CHEILARIS AND GÉZA TÓTH: Graph unique-maximum and conflict-free colorings. *J. Discrete Algorithms*, 9(3):241–251, 2011. [doi:10.1016/j.jda.2011.03.005] 3

[9] EDEN CHLAMTAC AND GYANIT SINGH: Improved approximation guarantees through higher levels of SDP hierarchies. In *Proc. 11th International Workshop on Approximation, Randomization and Combinatorial Optimization (APPROX'08)*, volume 5171 of *Lecture Notes in Computer Science*, pp. 49–62. Springer, 2008. [doi:10.1007/978-3-540-85363-3_5] 2

[10] LORENZO CIARDO, MARCIN KOZIK, ANDREI KROKHIN, TAMIO-VESA NAKAJIMA, AND STANISLAV ŽIVNÝ: 1-in-3 vs. Not-All-Equal: Dichotomy of a broken promise. *ACM Trans. Comput. Logic*, 26(2), April 2025. [doi:10.1145/3719007, arXiv:2302.03456] 3

[11] 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] 2

[12] G. EVEN, Z. LOTKER, D. RON, AND S. SMORODINSKY: Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pp. 691–700, 2002. [doi:10.1109/SFCS.2002.1181994] 3

[13] MAREK FILAKOVSKÝ, TAMIO-VESA NAKAJIMA, JAKUB OPRŠAL, GIANLUCA TASINATO, AND ULI WAGNER: Hardness of Linearly Ordered 4-Colouring of 3-Colourable 3-Uniform Hypergraphs. In *Proc. 41st International Symposium on Theoretical Aspects of Computer Science (STACS'24)*, volume 289 of *LIPIcs*, pp. 34:1–34:19, 2024. [doi:10.4230/LIPIcs.STACS.2024.34, arXiv:2312.12981] 2

[14] JOHAN HÅSTAD, BJÖRN MARTINSSON, TAMIO-VESA NAKAJIMA, AND STANISLAV ŽIVNÝ: A logarithmic approximation of linearly-ordered colourings. In *Proc. 27th International Workshop on Approximation, Randomization and Combinatorial Optimization (APPROX'24)*, volume 317 of *LIPIcs*, pp. 7:1–7:6, 2024. [doi:10.4230/LIPIcs.APPROX/RANDOM.2024.7] 2

[15] KEN-ICHI KAWARABAYASHI, MIKKEL THORUP, AND HIROTAKA YONEDA: Better coloring of 3-Colorable graphs. In *Proc. 56th Annual ACM Symposium on Theory of Computing (STOC'24)*, p. 331–339. Association for Computing Machinery, 2024. [doi:10.1145/3618260.3649768] 2

[16] MICHAEL KRIVELEVICH, RAM NATHANIEL, AND BENNY SUDAKOV: Approximating coloring and maximum independent sets in 3-uniform hypergraphs. *J. Algorithms*, 41(1):99–113, 2001. [doi:10.1006/jagm.2001.1173] 2

[17] MICHAEL KRIVELEVICH AND BENNY SUDAKOV: Approximate coloring of uniform hypergraphs. *J. Algorithms*, 49(1):2–12, 2003. [doi:10.1016/S0196-6774(03)00077-4] 2

[18] ANAND LOUIS, ALANTHA NEWMAN, AND ARKA RAY: Improved Linearly Ordered Colorings of Hypergraphs via SDP Rounding. In *Proc. 44th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'24)*, volume 323 of *LIPIcs*, pp. 30:1–30:19, 2024. [doi:10.4230/LIPIcs.FSTTCS.2024.30, arXiv:2405.00427] 2, 6

[19] TAMIO-VESA NAKAJIMA AND STANISLAV ŽIVNÝ: Linearly Ordered Colourings of Hypergraphs. In *Proc. 49th International Colloquium on Automata, Languages, and Programming (ICALP'22)*, volume 229 of *LIPIcs*, pp. 128:1–128:18, 2022. [doi:10.4230/LIPIcs.ICALP.2022.128] 2

[20] TAMIO-VESA NAKAJIMA AND STANISLAV ŽIVNÝ: Linearly Ordered Colourings of Hypergraphs. *ACM Trans. Comput. Theory*, 13(3–4), 2023. [doi:10.1145/3570909, arXiv:2204.05628] 2, 3, 6, 8

[21] SHAKHAR SMORODINSKY: *Combinatorial problems in computational geometry*. Ph. D. thesis, Tel-Aviv University, 2003. 3

[22] SHAKHAR SMORODINSKY: On the chromatic number of geometric hypergraphs. *SIAM J. Discrete Math.*, 21(3):676–687, 2007. [doi:10.1137/050642368] 3

[23] SHAKHAR SMORODINSKY: *Conflict-Free Coloring and its Applications*, pp. 331–389. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. [doi:10.1007/978-3-642-41498-5_12] 3

AUTHORS

Johan Håstad
Professor
Mathematics Department
KTH
Stockholm, Sweden
johanh@kth.se
https://www.csc.kth.se/~johanh/

Björn Martinsson
Ph. D. Student
Mathematics Department
KTH
Stockholm, Sweden
bmart@kth.se

Tamio-Vesa Nakajima
DPhil Student
Department of Computer Science
University of Oxford
Oxford, Oxfordshire, UK
tamio-vesa.nakajima@ox.ac.uk
https://www.tamionv.ro

Stanislav Živný
Professor
Department of Computer Science
University of Oxford
Oxford, Oxfordshire, UK
standa.zivny@ox.ac.uk
https://www.cs.ox.ac.uk/standa.zivny/

ABOUT THE AUTHORS

JOHAN HÅSTAD received his Bachelor of Science from Stockholm University in 1981, his Master of Science from Uppsala University in 1984, and his Ph.D. from MIT in 1986 under the supervision of Shafi Goldwasser. Johan was appointed Associate Professor at the Royal Institute of Technology in Stockholm, Sweden in 1988 and advanced to the level of Professor in 1992. He was elected a member of the Swedish Royal Academy of Sciences in 2001. He has research interests within several subareas of Theory of Algorithms and Complexity Theory but has recently mainly focused on the approximability of NP-hard optimization problems.

BJÖRN MARTINSSON, a PhD student from KTH Royal Institute of Technology under the supervision of Johan Håstad and Per Austrin with an interest in approximation algorithms, inapproximability and (promise) constraint satisfaction problems. In his free time, he enjoys participating in competitive programming contests using Python. He also runs a Discord server with ~ 5000 members dedicated to discussions about competitive programming and algorithms in general.

TAMIO-VESA NAKAJIMA (who usually goes by just Tamio) is a DPhil student at the University of Oxford, under the supervision of Standa Živný, since 2021. Their work centres on Constraint Satisfaction Problems, in particular approximating them in various ways. They first started doing Computer Science because of Informatics Olympiads (in Romania). In their free time, they perform and compose classical music.

STANISLAV ŽIVNÝ, known as Standa, studied in Prague, Amsterdam, and Turku before obtaining his DPhil from the University of Oxford in 2009 under the supervision of Peter Jeavons. His work centres around the application of mathematics to the design of algorithms for, and understanding the computational complexity of, constraint satisfaction problems. His work has been supported by a Royal Society University Research Fellowship, an ERC Starting Grant, and an ERC Consolidator Grant. In his spare time, he likes to go hiking with his family and friends.