Robot Positioning Using Torus Packing for Multisets

Chung Shue Chen[1]
Nokia Bell Labs, Nozay, France

Peter Keevash
Mathematical Institute, University of Oxford, UK

Sean Kennedy
Nokia Bell Labs, Ottawa, Canada

Élie de Panafieu
Nokia Bell Labs, Nozay, France

Adrian Vetta
McGill University, Montreal, Canada

Abstract

We consider the design of a positioning system where a robot determines its position from local observations. This is a well-studied problem of considerable practical importance and mathematical interest. The dominant paradigm derives from the classical theory of de Bruijn sequences, where the robot has access to a window within a larger code and can determine its position if these windows are distinct. We propose an alternative model in which the robot has more limited observational powers, which we argue is more realistic in terms of engineering: the robot does not have access to the full pattern of colours (or letters) in the window, but only to the intensity of each colour (or the number of occurrences of each letter). This leads to a mathematically interesting problem with a different flavour to that arising in the classical paradigm, requiring new construction techniques. The parameters of our construction are optimal up to a constant factor, and computing the position requires only a constant number of arithmetic operations.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases Universal cycles, positioning systems, de Bruijn sequences

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.43

Category Track A: Algorithms, Complexity and Games

Related Version *Extended Version*: `https://arxiv.org/abs/2404.09981` [8]

Supplementary Material *Software*: `https://gitlab.com/depanafieuelie/torus-packing-for-multisets` [13], archived at `swh:1:dir:2eb591ca09fb8739e2135e17b5f595b7f25ed924`

Funding *Peter Keevash*: Supported by ERC Advanced Grant 883810.
Adrian Vetta: NSERC Discovery Grant 04191

Acknowledgements Part of the work was carried out at the Laboratory for Information, Networking and Communication Sciences (`https://www.lincs.fr`). We thank Dr. Paolo Baracca, Siu-Wai Ho, Kenneth Shum and many colleagues for their great support and discussions.

[1] Authors presented in alphabetical order.

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).
Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;
Article No. 43; pp. 43:1–43:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Consider a robot located on a grid of coloured squares that must determine its position after observing part of the grid through a fixed viewing window. The dominant paradigm for this problem derives from the mathematical theory of de Bruijn sequences, i.e. binary (or bi-coloured) cyclic sequences of length 2^n in which each binary sequence of length n appears exactly once as a subsequence of consecutive entries. Such a sequence can be used for positioning a robot in one dimension: the robot sees a viewing window of length n; this window induces a subsequence with a unique colour pattern; from this colour pattern the robot can then reconstruct its position in the sequence (if we disregard issues of computational efficiency and error correction). Generalisations of this idea to higher dimensions and related combinatorial structures have led to a rich mathematical theory; see Section 1.2.

However, while this theory is mathematically pleasing, we will argue in Section 1.3 that engineering constraints support a model in which the robot does not have access to the full colour pattern in the window, but must infer its position only knowing the intensity of each colour, that is, the multiset of colours. More precisely, given an $n \times n$ grid, a robot with an $m \times m$ viewing window, and a palette of k colours, our task is to colour the grid so that each possible location of the viewing windows produces a different multiset of colours. Furthermore, it will be mathematically more natural to undertake this task for a torus of side n rather than a grid, and also to generalise to an arbitrary dimension d.

▶ **Example 1.** The grid colouring illustrated in Figure 1 has dimension $d = 2$, size $n = 8$, window size $m = 4$, and $k = 3$ colours (red 0, green 1, and blue 2). No two 4×4 subsquares contain the same multiset of colours. For example, two viewing windows are shown with multisets that have multiplicities for $(0, 1, 2)$ equal to $(6, 2, 8)$ and $(3, 5, 8)$, respectively. Observe that the second window "*wraps around*" because the grid is considered as a torus.

2	2	2	2	2	1	2	1
2	2	2	2	1	2	1	2
2	2	2	2	2	1	2	2
2	2	2	2	1	2	1	2
0	2	0	2	0	1	0	1
2	0	2	0	1	0	1	0
0	2	0	2	0	1	0	2
2	0	2	2	1	0	1	2

2	2	2	2	2	1	2	1
2	2	2	2	1	2	1	2
2	2	2	2	2	1	2	2
2	2	2	2	1	2	1	2
0	2	0	2	0	1	0	1
2	0	2	0	1	0	1	0
0	2	0	2	0	1	0	2
2	0	2	2	1	0	1	2

2	2	2	2	2	1	2	1
2	2	2	2	1	2	1	2
2	2	2	2	2	1	2	2
2	2	2	2	1	2	1	2
0	2	0	2	0	1	0	1
2	0	2	0	1	0	1	0
0	2	0	2	0	1	0	2
2	0	2	2	1	0	1	2

Figure 1 A grid coloring of dimension $d = 2$, size $n = 8$, window size $m = 4$. Observe that the two windows depicted contain distinct multisets of colours.

Our question of study is, given m, d, k, what is the largest grid size n for which position reconstruction is possible? Unfortunately, even the one-dimensional version of this question is the subject of several unsolved problems, discussed in Section 1.2. However, for practical purposes one can be content to relax from finding the optimal n given m, d, k to a value that is optimal up to a constant factor, where we think of d and k as fixed and consider the asymptotics for large m and n. There is a clear information theoretic barrier (see Observation 10) at $n = \Theta_{k,d}(m^{k-1})$, where the subscripts indicate that k and d are constants. The main contribution of this paper is a construction that achieves this theoretical optimum up to a constant factor, and moreover has optimal computational efficiency, in that only a constant number of arithmetic operations are required to compute the location of the window from its multiset of colours. We implemented and tested this construction in Python [13].

1.1 Definitions and Results

Notations. All vectors, tuples and sequences are indexed starting at 0. Integer intervals are denoted with square brackets, such as $[0, n-1]$. Given a length d, the vector $\boldsymbol{e}_i$ has all coordinates equal to 0, except the ith, which is equal to 1. The value of $\mathbb{1}_{\text{condition}}$ is 1 if *condition* is satisfied, and 0 otherwise. We write $i \equiv j \bmod n$ when the integer $i-j$ is divisible by n. Tables are represented with row indices increasing from top to bottom and column indices increasing from left to right, both starting at 0.

We first present general definitions that will be useful for discussing the literature and presenting our construction, starting with *cycle packings* for the one-dimensional case, and continuing with their higher dimensional generalisation, *torus packings*. Then the central objects of this article, *grid colouring*, are formally defined as particular cases of torus packings. Our main result, Theorem 9, is a near-optimal construction for grid colourings. It will make use of *vector sum packings* (Definition 11), which are particular cases of cycle packings.

▶ **Definition 2.** *Given an alphabet $\mathcal{A}$, a size n, a window size m, and a function f on $\mathcal{A}^m$, an $(\mathcal{A}, f, n, m)$-**cycle packing** is a function $W : \mathbb{Z} \mapsto \mathcal{A}$ that satisfies*

- `Periodicity.` *For all $x \in \mathbb{Z}$ we have $W_x = W_{x+n}$.*
- `Injectivity.` *If $f(W_x, W_{x+1}, \ldots, W_{x+m-1}) = f(W_y, W_{y+1}, \ldots, W_{y+m-1})$, for any integers x and y, then $x \equiv y \bmod n$.*

Let us explain this formal definition. Consider a circle composed of n squares, each receiving a letter from $\mathcal{A}$. A robot located on this circle wants to recover its position. It makes a local measurement, function f of the letters in a window of size m around it. The injectivity condition ensures that two positions of the robot (and thus of the window) correspond to two distinct values of f. Thus the robot is always able to deduce its position. Observe that the form of function f matters as different functions induce different types of information that robot can extract from the viewing window. To illustrate this point, consider the following three examples of $(\mathcal{A}, f, n, m)$-cycle packings. These examples all have $\mathcal{A} = \{0, 1, 2\}$ and $m = 3$ but differ in their functions f (which in turn lead to varying sizes of n).

▶ **Example 3.** Take the basic case where f is the identity function. The robot thus receives the entire colour pattern in its viewing window. Thus a cycle packing corresponds to a sequence of length n where all the contiguous subsequences of length m have distinct colour patterns. With $\mathcal{A} = \{0, 1, 2\}$ and window size $m = 3$, we have that $(0, 1, 1, 1, 2, 1, 0, 1, 2, 0, 1, 0, 2, 1, 1, 0, 0, 2, 2, 1, 2, 2, 2, 0, 2, 0, 0)$ is a cycle-packing of size $n = 27$ as each possible string of length 3 appears at most once; thus, the injectivity property holds. In fact, each such string appears *exactly* once (*e.g.* the factor $(0, 0, 0) = f(0, 0, 0)$ appears by wrapping around), making it a de Bruijn sequence. This exactness property is not required for cycle packings, but when it is satisfied the cycle packing is called a *universal cycle*.

▶ **Example 4.** Recall our motivation is that the robot extracts the colour intensities rather than the entire colour pattern. So instead of the identity function, assume that f is the multiset counting function which simply counts the number of appearances of each colour in the viewing window. Now, for the alphabet $\mathcal{A} = \{0, 1, 2\}$ with window size $m = 3$, the sequence $(0, 1, 1, 1, 2, 2, 2, 0, 0)$ is a cycle packing of size $n = 9$ as every multiset appears at most once (*e.g.* the multiset $(3, 0, 0) = f(0, 0, 0)$ arises by wrapping around) and the injectivity property holds. However, this cycle packing is not a universal cycle as no viewing window contains all three colours, so the multiset $(1, 1, 1)$ does not appear.

▶ **Example 5.** Again take $\mathcal{A} = \{0, 1, 2\}$ with window size $m = 3$, but now let f be the summation function. We claim the sequence $(0, 0, 0, 2, 2, 2, 1)$ is a cycle packing of size $n = 7$. To see this, observe that $f(0, 0, 0) = 0 + 0 + 0 = 0$, $f(0, 0, 2) = 0 + 0 + 2 = 2$, etc. Continuing these calculations, the outputted sequence of sums is $(0, 2, 4, 6, 5, 3, 1)$. This has distinct entries, so injectivity is satisfied. The reader may query the relevance of the summation function. In fact, this example is a special case of vector sum packings (Definition 11) which, in turn, will play a critical role in the construction underlying our main theorem.

Of course, our interest lies in dimension $d > 1$, so we now extend the definition of cycle packings to higher dimensions.

▶ **Definition 6.** *Given an alphabet $\mathcal{A}$, a size n, a window size m, a dimension d, and a function f on $\mathcal{A}^{m^d}$, an $(\mathcal{A}, f, n, m, d)$-**torus packing** is a function $W : \mathbb{Z}^d \mapsto \mathcal{A}$ that satisfies*

- `Periodicity.` *For all $\boldsymbol{x} \in \mathbb{Z}^d$ and $j \in [0, d-1]$, we have $W_{\boldsymbol{x}} = W_{\boldsymbol{x}+n\boldsymbol{e}_j}$ where $\boldsymbol{e}_j$ denotes the vector with a 1 in position j and 0 elsewhere.*
- `Injectivity.` *For any $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{Z}^d$, if $f\left((W_{\boldsymbol{x}+\boldsymbol{c}})_{\boldsymbol{c}\in[0,m-1]^d}\right) = f\left((W_{\boldsymbol{y}+\boldsymbol{c}})_{\boldsymbol{c}\in[0,m-1]^d}\right)$, then $x_j \equiv y_j \bmod n$ for all j (where x_j denote the jth coordinate of the vector $\boldsymbol{x}$).*

Note the key distinction in this definition is that the viewing window shares the same higher dimension as the torus. Again, by the periodicity property, we can identify a torus packing W with its pattern $(W_{\boldsymbol{x}})_{\boldsymbol{x}\in[0,n-1]^d}$. We are now ready to define the central objects of this paper. These are grid colourings, a particular case of torus packing that correspond exactly to the robot position reconstruction problem.

▶ **Definition 7.** *An (n, m, d, k)-**grid colouring** is an $(\mathcal{A}, f, n, m, d)$-torus packing with $|\mathcal{A}| = k$ and f mapping any sequence to the multiset of its entries.*

▶ **Example 8.** Consider, again, our example in Figure 1. This is an $(\mathcal{A}, f, n, m, d)$-torus packing with $\mathcal{A} = \{0, 1, 2\}$, $n = 8$, $m = 4$, $d = 2$ and where f is the multiset counting function. Consequently, since $|\mathcal{A}| = 3$, this is an $(8, 4, 2, 3)$-grid colouring.

The following is our main result on grid colourings, implemented and tested in Python [13].

▶ **Theorem 9.** *Fix a dimension $d \geq 2$ and a number of colours k of the form $bd + 1$ for some $b \geq 1$. For any window size m multiple of $2(k-1)$, there is an (n, m, d, k)-grid colouring W (explicitly constructed in the proof) with*

$$n \sim C_k^{1/d} \cdot m^{k-1} \qquad \textit{where} \qquad C_k = \left(\frac{2}{k-1}\right)^{k-1}.$$

Furthermore, for any $\boldsymbol{x}$ in $\mathbb{Z}^d$, given the multiset of colours in $(W_{\boldsymbol{x}+\boldsymbol{c}})_{\boldsymbol{c}\in[0,m-1]^d}$ one can compute $\boldsymbol{x}$ mod n with $O_{k,d}(1)$ arithmetic operations.

Our construction in Theorem 9 of size $n = \Omega_{k,d}(m^{k-1})$ is optimal up to a multiplicative constant. This fact follows from the following observation, which is immediate from counting considerations (injectivity requires the number of possible colour multisets to be at least the number of windows that they must distinguish).

▶ **Observation 10.** *The parameters of any (n, m, d, k)-grid colouring satisfy the inequality*

$$n^d \leq \binom{m^d + k - 1}{k - 1}.$$

In particular, for fixed dimension d and number of colours k, as the window size m tends to infinity, we have

$$n \leq C_k'^{1/d} \cdot m^{k-1}(1 + \mathcal{O}(m^{-1})) \qquad \textit{with} \qquad C_k' = \frac{1}{(k-1)!}.$$

We conclude this section by discussing related work and providing technological justifications for our positioning model.

1.2 Related Work

The combinatorial structures associated with torus packings (Definition 6) have a rich mathematical literature, starting from a problem solved in the 19th century, independently rediscovered by de Bruijn and now known as *de Bruijn sequences* (see [12]). The generalisation to higher dimensions was independently considered in several papers starting from the 1960's (see [24]) and has developed an extensive literature (see *e.g.* [17, 18, 20]) under the name of *de Bruijn tori*. The extension to general combinatorial structures encoded by sequences as in Definition 6 was proposed by Chung, Graham and Diaconis [9], who considered the one-dimensional problem ("*universal cycles*") for a variety of combinatorial structures. These early formulations of the problem generally asked for optimal solutions in which every object in a given combinatorial class is realised exactly once; in the context of Definition 6 this corresponds to strengthening injectivity to bijectivity. However, for practical purposes one can be satisfied with approximately optimal solutions, and given the difficulty of finding optimal solutions there is also a substantial literature (see *e.g.* [3, 10, 11]) finding approximate solutions from the perspective of packing (injectivity) and covering (surjectivity).

For the multiset encoding problem considered in this paper, finding the exact optimum is an open problem even for the one-dimensional setting of universal cycles, considered by Knuth [21, Fascicle 3, Section 7.2.1.3, Problem 109]. Hurlbert *et al.* [19] construct universal cycles for multisets with particular parameters, and Blanca and Godbole [3] consider the problem when the multisets have bounded multiplicities. Furthermore, the known results only consider the opposite regime of parameters from those needed for our application: we consider small palettes of colours (*a.k.a.* alphabets) and a large window, whereas previous techniques in the literature only apply to small windows and large alphabets. This comment also applies to the problem replacing "multiset" by "set", which is perhaps even more natural mathematically, given that it can be viewed as a hypergraph version of the Euler tour problem (introduced by Euler in the 18th century). Following many partial results, an exact solution to this problem (for small windows and large alphabets) was found by Glock *et al.* [14], solving a conjecture from Chung *et al.* [9]. We are unaware of any results in the literature on multiset packing in one dimension (meaning maximizing the number of multisets that appear in a sequence, rather than looking for sequences that contain all possible multisets).

Moving on from the abstract mathematical problem, we now consider some of the computer science literature aimed towards the specific application of indoor positioning. Much of the early work was surveyed by Burns and Mitchell [5]. More recent literature (see *e.g.* [2, 4, 6, 25]) has emphasised two further conditions that do not appear in the mathematical formulation but are naturally desirable for practical implementations: namely (a) computational efficiency, and (b) robustness against measurement errors. These works bring a variety of techniques from coding theory to bear on the positioning problem, providing efficient positioning algorithms with error correction. However, while these algorithms make natural use of an existing toolkit and are conceptually pleasing, we will argue that they are not addressing the most appropriate formulation of the problem from the point of view of the target application of robot positioning, for which a new paradigm is needed.

1.3 Motivation and Engineering Aspects

The focus of this paper is theoretical, concerning near-optimal designs for elegant combinatorial structures, namely torus packings. However, as stated, our motivation derives from the problem of position reconstruction (where the dimension is 2 and the torus of is considered a square). The vast number of applications of localization and positioning have prompted the design of a plethora of systems; see [31] and references therein.

Given the great practical importance of positioning systems, it behoves us to justify our claim that the approporiate way to model them is with multiset counting functions. We do this in this section by summarising two systems where our code (and variants of it) will be useful.

Light-Based Positioning. Recently, visible light based positioning systems have gained much attention [23, 26, 31] for two key reasons: (i) there is a strong demand for accurate but low-cost solutions, and (ii) there have been breakthroughs in the energy consumption and life expectancy of light-emitting diodes (LEDs). Contrary to systems requiring several light emitters and relying on triangulation [26], systems based on universal torus packing (such as de Bruijn torus [1, 27, 28] and our construction) rely on only one light source, reducing cost and energy consumption. Consider a room lit by a light-emitting diode (LED). A robot moving on the floor wears a light sensor or camera. A printed film is placed on top of the robot, above this sensor. The film is printed with a coloured grid, distorted so that depending on the position of the robot in the room, the light coming from the LED projects a square window of the coloured grid on the sensor [16]. If the coloured grid code depends on the respective positions of the colours (such as de Bruijn torus), the light detector must be a camera [22, 30] to recognize the pattern. If a torus packing for multisets is used instead, the robot can wear a simple light intensity sensor to recover the multiset of colours: the pattern of the colours is not needed, and we side-step the problematic issue of resolving the image. Such a device is considerably less expensive and has much shorter response time (lower latency) than a camera. It should be noted that the total light intensity (number of coloured rays) received by the detector depends on the position and also the orientation of the robot. Thus, for practical application of the construction presented in this article some redundancy should be added to allow for a unique position decoding in all cases. Positioning systems based on a de Bruijn torus also require redundancy to correct errors [6] and allow different orientations of the receiver [29]. We plan to explore redundancy in universal torus packing for multisets in future work.

Ambient Backscatters. An ambient backscatter is a small and inexpensive device that, upon reception of a radio wave, turns it into electricity and sends back data through a radio signal. Consider a warehouse where ambient backscatter devices are regularly placed, forming a grid. A robot with a radio emitter and receptor needs to locate itself in the warehouse. Each backscatter device, when in reach of the emitted radio wave, sends back its identification number (ID). If all devices have distinct IDs, the robot can determine its position based on the signals it receives. However, in the interest of energy consumption, IDs composed of few bits are preferred [7]. The same ID can be reused by several devices, provided that at any position, the multiset of IDs detected by the robot is unique. The relative position of the robot to its neighbouring backscatter devices is unknown, so the IDs should be chosen following a universal torus packing for multisets (a de Bruijn torus would require the robot to have directional antennas, making the system less practical and more expensive). For this application, our construction should be adapted to account for radial symmetry and decay of signal power in the detection window.

2 Overview

We have reduced the position reconstruction problem to the design of grid colourings. Consequently our main result, Theorem 9, is based upon an near-optimal grid colouring construction. A key building block in this construction will be the following family of cycle packings, which we call vector sum packings.

▶ **Definition 11.** *Given positive integers n, m, b and s, an (n, m, b, s)-**vector sum packing** is an $(\mathcal{A}, f, n, m)$-cycle packing with $\mathcal{A} = [0, s]^b$ and $f(\boldsymbol{z}_0, \dots, \boldsymbol{z}_{m-1}) = \boldsymbol{z}_0 + \cdots + \boldsymbol{z}_{m-1}$.*

▶ **Example 12.** For the special case of *vector dimension* $b = 1$ we have already encountered a $(8, 3, 1, 2)$-vector sum packing in Example 5.

▶ **Example 13.** Let us see an example of a vector sum packing with vector dimension $b = 3$. Set $m = 2$, $s = 1$, $n = 8$ and $(\boldsymbol{z}_0, \boldsymbol{z}_1, \dots, \boldsymbol{z}_7) = \left(\left(\begin{smallmatrix}0\\0\\0\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\0\\0\end{smallmatrix}\right), \left(\begin{smallmatrix}0\\1\\0\end{smallmatrix}\right), \left(\begin{smallmatrix}0\\0\\0\end{smallmatrix}\right), \left(\begin{smallmatrix}0\\0\\1\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\0\\0\end{smallmatrix}\right), \left(\begin{smallmatrix}0\\1\\1\end{smallmatrix}\right), \left(\begin{smallmatrix}0\\0\\0\end{smallmatrix}\right)\right)$. Then $f(\boldsymbol{z}_0, \boldsymbol{z}_1) = \left(\begin{smallmatrix}1\\0\\0\end{smallmatrix}\right)$, $f(\boldsymbol{z}_1, \boldsymbol{z}_2) = \left(\begin{smallmatrix}1\\1\\0\end{smallmatrix}\right)$, $f(\boldsymbol{z}_2, \boldsymbol{z}_3) = \left(\begin{smallmatrix}0\\1\\0\end{smallmatrix}\right)$, $f(\boldsymbol{z}_3, \boldsymbol{z}_4) = \left(\begin{smallmatrix}0\\0\\1\end{smallmatrix}\right)$, $f(\boldsymbol{z}_4, \boldsymbol{z}_5) = \left(\begin{smallmatrix}1\\0\\1\end{smallmatrix}\right)$, $f(\boldsymbol{z}_5, \boldsymbol{z}_6) = \left(\begin{smallmatrix}1\\1\\1\end{smallmatrix}\right)$, $f(\boldsymbol{z}_6, \boldsymbol{z}_7) = \left(\begin{smallmatrix}0\\1\\1\end{smallmatrix}\right)$, and $f(\boldsymbol{z}_7, \boldsymbol{z}_0) = \left(\begin{smallmatrix}0\\0\\0\end{smallmatrix}\right)$. As these sums all differ, the injectivity property holds and we have a vector sum packing.

The strategy of our construction is encapsulated in the following two key lemmas (proven in Section 3 and Section 4, respectively). The first reduces the construction of a grid colouring to the construction of a vector sum packing.

▶ **Lemma 14.** *Consider a dimension d, size n, window size m, and number of colours k, and assume the existence of integers b and s satisfying $k = bd + 1$ and $s = \frac{m^{d-1}}{bd}$. Then the existence of an (n, m, b, s)-vector sum packing implies the existence of a (n, m, d, k)-grid colouring.*

Given Lemma 14, the final piece in the puzzle is a construction of vector sum packings.

▶ **Lemma 15.** *For any $s \geq 1$, $m \geq 2$ and $b \geq 1$, there exists an $(n_b, m, b, 2s)$-vector sum packing with*

$$n_b = (2ms + 1)^{b-1} \left(2ms - \frac{1}{s}\right) + \frac{1}{s}.$$

To deduce the existence of the construction for Theorem 9, we combine Lemmas 14 and 15, choosing m as a large multiple of $2bd$, with $k = bd + 1$ and $s = \frac{m^{d-1}}{2bd}$. Then the value of n_b in Lemma 15 satisfies $n_b \sim \left(\frac{2}{k-1}\right)^{(k-1)/d} m^{k-1}$ for large m. We will prove in Corollary 22 that the position can be computed with a constant number of arithmetic operations.

We remark that the vector sum packing problem is related to the combinatorial theory of *antimagic labellings* (a natural variant on the classic topic of *magic labellings*) in which one is required to label the edges (or vertices) of a graph (or hypergraph) from a given set of integers (or vectors) so that vertices (or edges) are uniquely determined by the sum of their incident labels. A well-known conjecture of Ringel (cited in [15]) states that for any connected graph with $m > 1$ edges there is an antimagic labelling of its edges by $\{1, \dots, m\}$. It appears that this connection has not previously been exploited and may be fruitful for further research.

3 From Vector Sum Packing to Grid Colouring

The aim of this section is to prove Lemma 14, which reduces the grid colouring problem to the vector sum packing problem.

3.1 The Separation Property

In this section, we consider a fixed dimension $d \geq 2$, window size $m \geq 2$ and grid size $n \geq m$. For $\boldsymbol{x} \in \mathbb{Z}^d$, let us define the *window* of corner $\boldsymbol{x}$ as the set of points

$$\mathrm{Window}(\boldsymbol{x}) = \{\boldsymbol{x} + \boldsymbol{c} \mid \boldsymbol{c} \in [0, m-1]^d\}.$$

For the grid colouring problem, we require that the coordinates of each point $\boldsymbol{x}$ are uniquely determined modulo n by the multiset of colours of the points from $\mathrm{Window}(\boldsymbol{x})$. This multiset is denoted by $\mathrm{colourMultiset}(\boldsymbol{x})$, and the colour of the point $\boldsymbol{x}$ is denoted by $\mathrm{colour}(\boldsymbol{x})$.

We now present a sufficient condition, called *separation*, that guarantees this property. Each colour is represented as a pair (pigment, shade). We divide the set of colours into d pigment classes, one pigment class C_i for each dimension $i \in [0, d-1]$. Each pigment class contains b shades. Thus $C_i = \{c_{i,0}, c_{i,1}, \ldots, c_{i,b-1}\}$ where if i is the green pigment, say, then $c_{i,0}$ represents very light green and $c_{i,b-1}$ represents very dark green. The idea now is that for each square $\boldsymbol{x} \in \mathbb{Z}^d$, the coordinate x_i modulo n is uniquely determined by the pigment class C_i, for each $i \in [0, d-1]$, precisely by $\mathrm{colourMultiset}(\boldsymbol{x}) \cap C_i$. In particular, x_i is independent of any other pigment class $j \neq i$ on $\mathrm{Window}(\boldsymbol{x})$, for example, shades of red. Evidently, this separation property implies that if $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same colour multiset for every pigment class then $x_i \equiv y_i \bmod n$ for each $i \in [0, d-1]$ and, hence, $\boldsymbol{x} \equiv \boldsymbol{y} \bmod n$. Thus the separation property implies that we have a proper grid colouring.

Rather than working directly with separation, it will be convenient to consider the following two further conditions that together clearly imply separation.

`Dimensional Inconsistency`: Given $\boldsymbol{x}$ and $\boldsymbol{y}$, if $x_i \neq y_i \bmod n$ then

$$\mathrm{colourMultiset}(\boldsymbol{x}) \cap C_i \neq \mathrm{colourMultiset}(\boldsymbol{y}) \cap C_i.$$

`Anti-Dimensional Consistency`: Given $\boldsymbol{x}$ and $\boldsymbol{y}$, if $x_i = y_i \bmod n$ then

$$\mathrm{colourMultiset}(\boldsymbol{x}) \cap C_i = \mathrm{colourMultiset}(\boldsymbol{y}) \cap C_i.$$

To ensure anti-dimensional consistency it will be convenient to focus on the following condition called *quasi-periodicity*, which states that if some $\boldsymbol{x}$ is coloured a shade of pigment i then any translate $\boldsymbol{y}$ of $\boldsymbol{x}$ by distance m in any dimension other than i has the same colour as $\boldsymbol{x}$

$$\forall i \neq j, \text{ if } \mathrm{colour}(\boldsymbol{x}) \in C_i \text{ then } \mathrm{colour}(\boldsymbol{x} + m\, \boldsymbol{e}_j) = \mathrm{colour}(\boldsymbol{x}).$$

▶ **Lemma 16.** *Any quasi-periodic grid colouring satisfies anti-dimensional consistency.*

Proof. Take a quasi-periodic colouring. It suffices to show, given x_i, that $\mathrm{colourMultiset}(\boldsymbol{x}) \cap C_i$ is fixed. This will hold if the number of squares of colour $c_{i,\ell}$ in a window does not change if we translate by one square in any dimension j other than i. Let us define

$$A = \mathrm{Window}(\boldsymbol{x}) \setminus \mathrm{Window}(\boldsymbol{x} + \boldsymbol{e}_j), \qquad B = \mathrm{Window}(\boldsymbol{x} + \boldsymbol{e}_j) \setminus \mathrm{Window}(\boldsymbol{x}).$$

Since B is obtained from A by translation by $m\boldsymbol{e}_j$, quasi-periodicity implies that the number of points of colour $c_{i,\ell}$ in A and B are equal. Thus, the number of points of colour $c_{i,\ell}$ in $\mathrm{Window}(\boldsymbol{x})$ and $\mathrm{Window}(\boldsymbol{x} + \boldsymbol{e}_j)$ are equal as well. As this argument applies for any $j \neq i$, the anti-dimensional consistency property holds. Thus, quasi-periodicity implies anti-dimensional consistency. ◀

3.2 Separation via Vector Sum Packing

Assume we have an (n, m, b, s)-vector sum packing $(\boldsymbol{z}_j)_{j\in\mathbb{Z}}$ of size n and window size m, containing vectors in $[0, s]^b$, where s is a positive integer equal to $\frac{m^{d-1}}{bd}$ for some dimension $d \geq 2$. We will now construct a (n, m, d, k)-grid colouring with number of colours $k = bd + 1$. We detail our algorithm below and illustrate it in Figure 2. As we saw in the previous section, to ensure the separation condition, it is sufficient that our grid colouring satisfies dimensional inconsistency and quasi-periodicity.

(a) We begin with an initial colouring of the grid using only d pigments, each coming in b different shades. Take a pigment $i \in [0, d-1]$ and shade $h \in [0, b-1]$. Then the point $\boldsymbol{x} = (x_0, \dots, x_{d-1})$ has pigment i and shade h if and only if

$$\sum_{j=0}^{d-1} x_j \equiv i + hd \bmod bd.$$

This initial colouring is quasi-periodic. But, of course, it does not satisfy dimensional inconsistency. To rectify this, we will apply the last unused colour, which we call *blank*, to erase some colours from the initial colouring.

(b) Consider a dimension $i \in [0, d-1]$. We associate to it the pigment i and the set C_i of the corresponding b shades. For each $j \in \mathbb{Z}$, let

$$B_{i,j} = \{\boldsymbol{x} \mid x_i = j \text{ and } \forall \ell \neq i,\ x_\ell \in [0, m-1]\}.$$

We apply the blank colour to erase some of the colours from $B_{i,j}$, so that the number of points of shade $c_{i,\ell}$ is equal to the ℓth component of $\boldsymbol{z}_j$. This is always possible, because in the initial colouring, $B_{i,j}$ contains $s = \frac{m^{d-1}}{bd}$ occurrences of shade $c_{i,\ell}$, and the values of the vectors from the vector sum packing are all in $[0, s]$.

(c) We apply quasi-periodicity to reproduce this construction on the rest of the grid. Specifically, point $\boldsymbol{x} = (x_0, \dots, x_{d-1})$ of pigment i and shade h in the initial colouring is erased if and only if the point $\boldsymbol{y}$ with $y_i = x_i$ and for all $j \neq i$, $y_j = x_j \bmod m$ was erased at step (b).

The injectivity of the vector sum packing implies the dimensional inconsistency of our grid colouring. By construction, our grid colouring is quasi-periodic, so by Lemma 16, it satisfies the separation property, concluding the proof of Lemma 14.

3.3 An Example

An illustration of the proof of Lemma 14 is given in Figure 2 on a grid of dimension $d = 2$, size $n = 8$, window size $m = 4$, number of colours $k = 3$. To create the grid-colouring we use the vector sum packing $(\boldsymbol{z}_0, \boldsymbol{z}_1, \dots, \boldsymbol{z}_7) = (0, 0, 0, 0, 2, 2, 2, 1)$ (note each vector has dimension 1, so is represented by its content). In particular, $s = 2$ and $b = 1$; note that $k = bd + 1$ and $s = \frac{m^{d-1}}{bd}$. Recall, the aim is that number of occurrences of 0 (*resp.* 1) in a 4 by 4 square characterizes its row (*resp.* column) number. Following the proof of Lemma 14, we start with a periodic colouring, represented in (a). We now use the blank colour 2 to erase some of the 0. First, in (b), we erase entries in the first 4 columns so the number of occurrences of 0 in these columns for the eight rows are $(0, 0, 0, 0, 2, 2, 2, 1)$. Second in (c), we apply quasi-periodicity on the rest of the grid. Next, in (d) and (e), we apply the same approach to erase some of the 1. The final result is (e). No two 4×4 subsquares contain the same multiset of colours.

Let us now illustrate how this grid is used for localization. Recall that our convention is to number the rows from top to botton, and column from left to right, both starting at 0. Assume we are measuring in a window (*i.e.* a 4×4 subsquare) the multiset of colors

0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0

(a)

2	1	2	1	0	1	0	1
1	2	1	2	1	0	1	0
2	1	2	1	0	1	0	1
1	2	1	2	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	2	1	0	1	0

(b)

2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	2	1	0	1	2

(c)

2	2	2	2	2	1	2	1
2	2	2	2	1	2	1	2
2	2	2	2	2	1	2	2
2	2	2	2	1	2	1	2
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	2	1	0	1	2

(d)

2	2	2	2	2	1	2	1
2	2	2	2	1	2	1	2
2	2	2	2	2	1	2	2
2	2	2	2	1	2	1	2
0	2	0	2	0	1	0	1
2	0	2	0	1	0	1	0
0	2	0	2	0	1	0	2
2	0	2	2	1	0	1	2

(e)

Figure 2 Illustration of the steps of the proof of Lemma 14 on a grid of dimension $d = 2$, size $n = 8$, window size $m = 4$, number of colours $k = 3$.

containing 5 occurrences of 0, 3 occurrences of 1 and 8 occurrences of 2. We wish to locate this window in the grid (e). The naive algorithm is to consider each possible window in the grid and compare the multisets of colors. This becomes costly for large grids, so we present a more efficient algorithm. By convention, color 0 is used to determine the row. Looking at the vector sum packing $(0, 0, 0, 0, 2, 2, 2, 1)$, we observe that the sequence whose jth element is the sum of $m = 4$ consecutive elements starting at position j, is $(0, 2, 4, 6, 7, 5, 3, 1)$. The number 5 is located at position 5 in this sequence, so the upper-left corner of the window we are seeking has row number 5. To determine the column, we consider the color 1. Its number of occurrence 3 has position 6 in $(0, 2, 4, 6, 7, 5, 3, 1)$, so the column number is 6. On the torus (e), the 4×4 subsquare with top left corner in row 5 and column 6 indeed contains 5 occurrences of 0, 3 occurrences of 1 and 8 occurrences of 2.

Observe that for any fixed dimension, the localization problem in the grid reduces in constant complexity to the problem of computing the position of a vector in a vector sum packing. We call this second problem *decoding*. In the next section, we will present our construction for vector sum packings, as well as a decoding algorithm with constant complexity (in the number of arithmetic operations, as the dimension d and parameter b, defined there, are fixed).

A grid colouring of size 256, window size 8, and 5 colours is presented in the appendix.

4 Vector Sum Packing

This section presents our construction of vector sum packings, thus proving Lemma 15, which is the last missing ingredient for the proof of our main result Theorem 9.

4.1 Profiles and Duals

To build vector sum packings (see Definition 11), we introduce certain integer sequences that we call *profiles*. Profiles with different parameters will be used to fill the coordinates of the vector sum packing, in Lemma 21.

The *m-dual* of a profile w is an integer sequence of same length $|w|$, defined as the sum of the elements of w on a cycling window of length m

$$\mathrm{Dual}_m(w) = \left(\sum_{j=i}^{i+m-1} w_{i \bmod |w|} \right)_{i \in [0,|w|-1]}.$$

Consider integers $s \geq 1$ and $m \geq 2$. Let us write sequences of length 1 as (σ) and sequences of length m as $(\sigma_0, \sigma_1, \ldots, \sigma_{m-1})$. For a finite sequence L and a nonnegative integer T, the sequence obtained by concatenating T copies of L one after the other is denoted by L^T. Let $\emptyset$ denote the emptysequence, and let $a \cdot b$ denote the concatenation of the sequences a and b, so $(1,2,3) \cdot (4)$ is equal to $(1,2,3,4)$. In the following tables, the rows and columns are numbered starting at 0, in red. Straight lines have been added between some of the rows to distinguish parts of the tables following different rules.

Let us define the sequence $\mathrm{Profile}(s,m,0)$ as the concatenation of the cells from the following table, read line by line from top left to bottom right.

	0	1	2	$\cdots$	$s-1$
0	$(0,\ldots,0)$	$(2,\ldots,2)$	$(4,\ldots,4)$	$\cdots$	$(2s-2,\ldots,2s-2)$
1	$(2s,\ldots,2s,2s-1)$	$(2s-2,\ldots,2s-2,2s-3)$	$(2s-4,\ldots,2s-4,2s-5)$	$\cdots$	$(2,\ldots,2,1)$

For example, we have $\mathrm{Profile}(1,3,0) = (0,0,0,2,2,1)$ and $\mathrm{Profile}(2,2,0) = (0,0,2,2,4,3,2,1)$.

▶ **Lemma 17.** $\mathrm{Profile}(s,m,0)$ *has length* $2ms$. *Furthermore, its* m*-dual is*

$$\mathrm{Dual}_m(\mathrm{Profile}(s,m,0)) = (0,2,4,\ldots,2ms-2,2ms-1,2ms-3,2ms-5,\ldots,1).$$

Proof. $\mathrm{Profile}(s,m,0)$ has length $2ms$ because each entry in the table is a sequence of cardinality m and there are $2s$ entries in the table. The reader may easily verify that the m-dual begins with non-negative even numbers increasing up to $2ms-2$ followed by positive odd numbers decreasing down from $2ms-2$. Thus, the m-dual contains every integer in $[0, 2m-1]$ exactly once. ◀

Let us define the $(s,m,0)$-decoding function as the function that associates to an integer v its smallest index in the m-dual of $\mathrm{Profile}(s,m,0)$. For example, the $(2,2,0)$-decoding function sends 6 to 3, because the 2-dual of $\mathrm{Profile}(2,2,0) = (0,0,2,2,4,3,2,1)$ is $(0,2,4,6,7,5,3,1)$, where the first (and only) occurrence of 6 is at position 3.

▶ **Corollary 18.** *The* $(s,m,0)$*-decoding function is*

$$v \mapsto \begin{cases} \frac{v}{2} & \text{if } v \text{ is even,} \\ 2ms-1-\frac{v-1}{2} & \text{if } v \text{ is odd.} \end{cases}$$

It is computable in a constant number of arithmetic operations.

For any positive integer T and $m \geq 2$, let us define the sequence $\mathrm{Profile}(s, m, T)$ as the concatenation of the cells from the following table.

	0	1	2	$\cdots$	$s-1$
0	$(0,\dots,0,0)^T$	$(0,\dots,0,2)^T$	$(0,\dots,0,4)^T$	$\cdots$	$(0,\dots,0,2s-2)^T$
1	$(0,\dots,0,0,2s)^T$	$(0,\dots,0,2,2s)^T$	$(0,\dots,0,4,2s)^T$	$\cdots$	$(0,\dots,0,2s-2,2s)^T$
$\vdots$	$\vdots$	$\vdots$			$\vdots$
$m-1$	$(0,2s,\dots,2s,2s)^T$	$(2,2s,\dots,2s,2s)^T$	$\cdots$	$\cdots$	$(2s-2,2s,\dots,2s)^T$
m	$(2s)^{mT-1}$	$(2s-1,2s,\dots,2s)^T$	$(2s-3,2s,\dots,2s)^T$	$\cdots$	$(3,2s,\dots,2s)^T$
$m+1$	$(1,2s,2s,\dots,2s)^{T-1}$	$(1,2s,\dots,2s,2s-2)^T$	$(1,2s,\dots,2s,2s-4)^T$	$\cdots$	$(1,2s,\dots,2s,2)^T$
$m+2$	$(1,2s,\dots,2s,0)^T$	$(1,2s,\dots,2s-2,0)^T$	$(1,2s,\dots,2s,2s-4,0)^T$	$\cdots$	$(1,2s,\dots,2s,2,0)^T$
$\vdots$	$\vdots$	$\vdots$			$\vdots$
$2m-1$	$(1,2s,0,\dots,0)^T$	$(1,2s-2,0,\dots,0)^T$	$\cdots$	$\cdots$	$(1,2,0,\dots,0)^T$
$2m$	$(1,0,\dots,0)^{T-1}\cdot(1)$	$\emptyset$	$\emptyset$	$\cdots$	$\emptyset$

▶ **Lemma 19.** *For any $m \geq 2$, $\mathrm{Profile}(s, m, T)$ has length $m((2ms + 1)T - 2)$. Furthermore, its m-dual is obtained by concatenation of the cells of the following table*

	0	1	2	$\cdots$	$s-1$
0	$(0)^{mT}$	$(2)^{mT}$	$(4)^{mT}$	$\cdots$	$(2s-2)^{mT}$
1	$(2s)^{mT-1}$	$(2s+2)^{mT}$	$(2s+4)^{mT}$	$\cdots$	$(4s-2)^{mT}$
2	$(4s)^{mT-1}$	$(4s+2)^{mT}$	$(4s+4)^{mT}$	$\cdots$	$(6s-2)^{mT}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
$m-1$	$(2(m-1)s)^{mT-1}$	$(2(m-1)s+2)^{mT}$	$(2(m-1)s+4)^{mT}$	$\cdots$	$(2ms-2)^{mT}$
m	$(2ms)^{mT-1}$	$(2ms-1)^{mT}$	$(2ms-3)^{mT}$	$\cdots$	$(2(m-1)s+3)^{mT}$
$m+1$	$(2(m-1)s+1)^{mT-1}$	$(2(m-1)s-1)^{mT}$	$(2(m-1)s-3)^{mT}$	$\cdots$	$(2(m-2)s+3)^{mT}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
$2m-1$	$(2s+1)^{mT-1}$	$(2s-1)^{mT}$	$(2s-3)^{mT}$	$\cdots$	$(3)^{mT}$
$2m$	$(1)^{mT-1}$	$\emptyset$	$\emptyset$	$\cdots$	$\emptyset$

Proof. In the table, there are $2ms - 2$ cells containing sequences of length mT, one cell containing a sequence of length $mT - 1$, one cell containing a sequence of length $m(T - 1)$ and one celle containing a sequence of length $m(T - 1) + 1$, so $\mathrm{Profile}(s, m, T)$ has length

$$(2ms - 2)mT + mT - 1 + m(T - 1) + m(T - 1) + 1 = mT(2ms + 1) - 2m$$

as desired. Again, the reader may verify that the m-dual begins with non-negative even numbers increasing up to $2ms$ followed by positive odd numbers decreasing down from $2ms - 1$ (repeated in the quantities specified). ◀

We define the (s, m, T)-decoding function as the function that associates to an integer v its smallest index in the m-dual of $\mathrm{Profile}(s, m, T)$.

▶ **Corollary 20.** *The (s, m, T)-decoding function output on the input v is computed using the following algorithm. If v is even, we define $r = \lfloor \frac{v}{2s} \rfloor$ and $c = \frac{v}{2} \bmod [s]$. They represent the row and column in the m-dual from Lemma 19. Then the output of the decoding function is*

$$(r\,m\,T\,s - r + 1)\mathbb{1}_{r>0} + (c\,m\,T - 1 + \mathbb{1}_{r=0})\mathbb{1}_{c>0}.$$

Otherwise, v is odd and we define $r = \lfloor \frac{2\,m\,s - v + 1}{2s} \rfloor$ and $c = \frac{2\,m\,s - v + 1}{2} \bmod s$. The output is then

$$1 + m(m\,T\,s - 1) + (m\,T\,s\,r - r)\mathbb{1}_{r>0} + (c\,m\,T - 1)\mathbb{1}_{c>0}.$$

It is computable in a constant number of arithmetic operations.

4.2 Generating a Vector Sum Packing

We will explicitly construct the vector sum packing by combining profiles, thus proving Lemma 15.

Recall that an (n, m, b, s)-vector sum packing is characterised by its pattern $(\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{n-1})$ where each $\boldsymbol{z}_i$ is a vector in $[0, s]^b$. These vectors will be defined as the columns of a matrix $M^{(b)}$. Furthermore, the rows of this matrix will be constructed using sequences given by the Profile(s, m, T).

The matrix $M^{(b)}$ will have b rows and $m \cdot T_b = n_b$ columns. Specifically, given $s \geq 1$ and $m \geq 2$, we set $T_0 = 0$ and $T_1 = 2s$. Then, for each $b \geq 1$, we recursively set

$$T_{b+1} = (2ms + 1)T_b - 2.$$

Thus we obtain the dimensions of our $M^{(b)}$ matrices. To fill in the entries of the matrices we again apply a recursive construction.

- For $b = 1$, the matrix $M^{(1)}$ has only one row, which is identical to the sequence Profile$(s, m, 0)$. We remark that $M^{(1)}$ does indeed have $n_1 = m \cdot T_1 = 2ms$ columns, as required by Lemma 17.
- Next consider the case $b \geq 2$. The basic idea is that $M^{(b)}$ should simply be the concatenation on $2ms + 1$ copies of $M^{(b-1)}$, plus an additional row identical to the sequence Profile(s, m, T_{b-1}), which will be used to distinguish between the different copies.

However, this basic idea does not scale correctly, so instead of concatenating identical copies of $M^{(b-1)}$, we also concatenate truncated copies of $M^{(b-1)}$. Specifically, we allow for the truncated matrix $M^{(b-1,\star)}$ which is identical to $M^{(b-1)}$ except that its first column is removed.

We now set $M^{(b)} = M^{(b-1,0)} \circ M^{(b-1,2)} \circ \cdots \circ M^{(b-1,2ms)} \circ M^{(b-1,2ms-1)} \circ \cdots \circ M^{(b-1,1)})$, where each $M^{(b-1,\ell)}$ is either $M^{(b-1)}$ or $M^{(b-1,\star)}$ and where $\circ$ denotes the concatenation operation. Thus, it remains, to prescribe, for each for $\ell \in [0, 2ms]$ whether $M^{(b-1,\ell)}$ is set equal to $M^{(b-1)}$ or $M^{(b-1,\star)}$. To do this, we use the m-dual of the Profile(s, m, T_{b-1}). In particular, define I_ℓ to be the set of indices where the m-dual of the profile takes value ℓ. That is,

$$I_\ell = \{i \mid \mathrm{Dual}_m(\mathrm{Profile}(s, m, T_{b-1}))_i = \ell\} \tag{1}$$

Recall that $n_b = mT_b = m \cdot ((2ms + 1) \cdot T_{b-1} - 2)$. Observe then, from Lemma 19, that the (I_ℓ)s are disjoint integer intervals of length either n_{b-1} or $n_{b-1} - 1$ whose union is $[0, n_b - 1]$. For each $\ell \in [0, 2ms]$, we now define

$$M^{(b-1,\ell)} = \begin{cases} M^{(b-1)} & \text{if } |I_\ell| = n_{b-1}, \\ M^{(b-1,\star)} & \text{if } |I_\ell| = n_{b-1} - 1. \end{cases}$$

The resultant construction of $M^{(b)}$ is then illustrated in Figure 3. For example, for $s = 1$, $m = 2$ and $b = 2$, we have $T_0 = 0$ and $T_1 = 2$, so

$$\begin{aligned} \mathrm{Profile}(s, m, T_{b-1}) &= \mathrm{Profile}(1, 2, 2) = (0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 2, 1, 2, 1, 0, 1), \\ \mathrm{Dual}_m(\mathrm{Profile}(s, m, T_{b-1})) &= (0, 0, 0, 0, 2, 2, 2, 4, 4, 4, 3, 3, 3, 1, 1, 1), \\ \mathrm{Profile}(s, m, T_{b-2}) &= \mathrm{Profile}(1, 2, 0) = (0, 0, 2, 1), \end{aligned}$$

so

$$M^{(2)} = \begin{pmatrix} 0 & 0 & 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 2 & 2 & 2 & 2 & 1 & 2 & 1 & 0 & 1 \end{pmatrix}.$$

$$M^{(b)} = \begin{array}{|c|c|c|c|c|c|} \hline \overbrace{M^{(b-1,0)}}^{I_0} & \overbrace{M^{(b-1,2)}}^{I_2} & \cdots & \overbrace{M^{(b-1,2ms)}}^{I_{2ms}} & \cdots & \overbrace{M^{(b-1,1)}}^{I_1} \\ \hline \multicolumn{6}{|c|}{\text{Profile}(s,m,T_{b-1})} \\ \hline \end{array}$$

Figure 3 The recursive construction of the matrix $M^{(b)}$. The indices j for the sets I_j on top of the figure are first increasing even numbers, then decreasing odd numbers.

As stated, we will take our vectors $(\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_b-1})$ to be the columns of $M^{(b)}$. That is, let $V_{i,j} = M^{(b)}_{i \bmod n_b, j}$. Then, for any $i \in \mathbb{Z}$, we have $\boldsymbol{z}_i = (V_{i,j})_{j\in[0,b-1]}$.

▶ **Lemma 21.** *The sequence of vectors $(\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_b-1})$ is the pattern of an $(n_b, m, b, 2s)$-vector sum packing.*

Proof. Given $(\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_b-1})$ recall that $f(\boldsymbol{z}_j, \boldsymbol{z}_{j+1}, \ldots, \boldsymbol{z}_{j+m-1}) = \boldsymbol{z}_j + \boldsymbol{z}_{j+1} + \cdots + \boldsymbol{z}_{j+m-1}$, where the indices are taken modulo n_b. Our task is to prove that f is injective.

We will show this by extending the definition of an m-dual to matrices: we define the m-dual of a matrix M with n columns to be the matrix D of the same dimensions where for all i, the ith column of D is equal to the sum of the columns of M of all indices in $[i, i+m-1]$ modulo n.

Proving $(\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_b-1})$ is an $(n_b, m, b, 2s)$-vector sum packing is then equivalent to proving that the m-dual of $M^{(b)}$ does not contain any identical columns. We will do so by induction on b.

Initialization. For the base case $b = 1$, recall that $M^{(b)} = \text{Profile}(s, m, 0)$. Hence, by Lemma 17, $\text{Dual}_m(M^{(b)})$ does not contain two identical columns.

Induction Step. For the induction hypothesis, assume that $\text{Dual}_m(M^{(b-1)})$ does not contain two identical columns. Now define $\text{Profile}^*(s, m, T)$ to be equal to $\text{Profile}(s, m, T)$ except with the first 0 removed. By construction, each row $j \in [0, b-1]$ of $M^{(b)}$ is a concatenation of the form $P_0 \circ P_1 \circ \cdots \circ P_r$ where each P_i is equal either to $\text{Profile}(s, m, T_j)$ or to $\text{Profile}^*(s, m, T_j)$. Next observe that both $\text{Profile}(s, m, T_j)$ and $\text{Profile}^*(s, m, T_j)$ start with $m-1$ occurrences of 0. This implies that m-dual and concatenation commutes as

$$\text{Dual}_m(P_0 \circ P_1 \circ \cdots \circ P_r) = \text{Dual}_m(P_0) \circ \text{Dual}_m(P_1) \circ \cdots \circ \text{Dual}_m(P_r).$$

Therefore the m-dual of the matrix $M^{(b)}$ is

$$\text{Dual}_m(M^{(b)}) = \begin{array}{|c|c|c|} \hline \overbrace{\text{Dual}_m(M^{(b-1,0)})}^{I_0} & \cdots & \overbrace{\text{Dual}_m(M^{(b-1,1)})}^{I_1} \\ \hline \multicolumn{3}{|c|}{\text{Dual}_m(\text{Profile}(s,m,T_{b-1}))} \\ \hline \end{array} \tag{2}$$

Assume that two columns c and c' with column indices i and i' in $\text{Dual}_m(M^{(b)})$ are equal. Let $\ell \in [0, 2ms]$ denote the value of c in its final row $b-1$. Thus, c' also has entry ℓ in its last row. But the last row $b-1$ of $\text{Dual}_m(M^{(b)})$ is defined to equal $\text{Dual}_m(\text{Profile}(s, m, T_{b-1}))$. Hence, by definition of I_ℓ from Equation (1), we have $i \in I_\ell$ and $i' \in I_\ell$. Let d and d' be obtained from c and c' by removing their last row $b-1$. By Equation (2), both d and d' belong to $\mathrm{Dual}_m(M^{(b-1,\ell)})$. Thus, they both belong to $\mathrm{Dual}_m(M^{(b-1)})$. By the induction hypothesis, this implies $i = i'$ and concludes the proof. ◀

Let the (n, m, b, s)-decoding function for vector sum packing be defined as the function that inputs an integer vector $\boldsymbol{x}$ and outputs its index in the m-dual of the (n, m, b, s)-vector sum packing we defined.

▶ **Corollary 22.** *For fixed b, the $(n_b, m, b, 2s)$-decoding function for our vector sum packing is computable in a constant number of arithmetic operations.*

Proof. This decoding function is computed recursively, following the recursive construction of matrix $\mathrm{Dual}_m(M^{(b)})$ from Equation (2). It inputs a vector $\boldsymbol{x}$ and an auxiliary Boolean parameter B, initialized at *False*, that indicates if we are looking for localization in a vector sum packing where the first vector has been removed.

We first use the decoding function for profiles from Corollaries 18 and 20 on the last coordinate of $\boldsymbol{x}$ to compute an integer c, and set $p = c$ if B is equal to *False*, and $p = c - 1$ if B is equal to *True*. For $b = 1$, as $\boldsymbol{x}$ is a vector of dimension 1, the algorithm stops and p is returned. Otherwise, by construction, p is the smallest possible index for any vector whose last coordinate is equal to the last coordinate of $\boldsymbol{x}$. Now, in Equation (2), we want to determine whether the matrix $\mathrm{Dual}_m(M^{b-1,j})$ on top of p is equal to $\mathrm{Dual}_m(M^{b-1})$ or to $\mathrm{Dual}_m(M^{b-1,\star})$. As explained in the construction, this is decided by looking at $\mathrm{Dual}_m(\mathrm{Profile}(s, m, T_{b-1}))$ from Lemma 19. In this sequence, let ℓ denote the number of repetition of the element at position p.

- If $\ell = mT_{b-1}$, then we are working with $\mathrm{Dual}_m(M^{b-1})$. In that case, we call recursively the $(n_{b-1}, m, b-1, 2s)$-decoding function on the vector $\boldsymbol{x}$ without its last coordinate, with auxiliary parameter equal to *False*. The output is added to p and returned.
- Otherwise, we have $\ell = mT_{b-1} - 1$, and we are working with $\mathrm{Dual}_m(M^{b-1,\star})$. We call recursively the $(n_{b-1}, m, b-1, 2s)$-decoding function on the vector $\boldsymbol{x}$ without its last coordinate, with auxiliary parameter equal to *True*. The output is added to p and returned.

Deciding whether $\ell = mT_{b-1}$ or $\ell = mT_{b-1} - 1$ is achieved by looking at the parity of p and its value modulo s. This recursive construction has depth b, so for b fixed, it requires only a constant number of arithmetic operations. Python code for this algorithm is provided in [13]. ◀

4.3 Summary and Complexity

Let us summarize our algorithm constructing a grid coloring. It inputs a dimension $d \geq 2$ and two other parameters $b \geq 1$ and $t \geq 1$. The first step is to compute the window size $m = 2bdt$, the number of colors $k = bd + 1$, the parameter $s = (2bd)^{d-2}t^{d-1}$ to ensure $2s = \frac{m^{d-1}}{bd}$, and the grid size $n = (2ms + 1)^{b-1}(2ms - 1/s) + 1/s$. The second step is to construct an $(n, m, b, 2s)$-vector sum packing, as described in Section 4.2, using the *profile* sequences defined in Section 4.1. In the third step, we finally colour the points of our grid of side n and dimension d. Our set of colours is $\{0, 1, \dots, k-1\}$. Here the last colour $k-1$ is a "*blank*" color used to erase the other colours. The other colours are divided into d sets, called pigment classes. Each pigment class contains b colours, which we call its *shades*. To each dimension of the grid is associated a unique pigment. The vector sum packing is then used, as explained in Section 3, to colour the points of the grid.

Next consider the localization problem. Localization means, given a multiset S of colours, the recovery of the unique window of size m in the grid that contains this multiset of colors (if it exists). To achieve localization, we proceed dimension by dimension. We count in S the colours from the pigment class corresponding to the dimension considered and make a vector out of it. For example, if the dimension corresponds to the colours $4, 5, 6$ and S contains three occurrences of the color 4, zero occurrences of the colour 5 and two occurrences of the colour 6, the vector is $(3, 0, 2)$. We use the decoding algorithm for vector sum packing from Corollary 22 to translate this vector into a coordinate. Having achieved this for every coordinate, we deduce the position of the window whose multiset of colours is equal to S.

The construction of the grid and localization procedure are illustrated in Section 3.3. We measure complexity as the number of arithmetic operations for b and d fixed, while t goes to infinity. The construction of the grid has complexity proportional to the size of its output, which is $\mathcal{O}(n^d) = \mathcal{O}(t^{b\,d^2})$. Corollary 22 implies that the complexity of the localization algorithm is constant.

5 Conclusion

Many interesting directions for future research remain, both theoretical and practical. One nice extension would be to make our grid colouring robust by allowing for error detection and correction. This has been achieved for other cycle and torus packing problems (see, for example, [2, 4, 6, 25]) and is a necessary step for practical applications. Another valuable contribution would be to study disc-like windows rather than the square windows examined in this article. This would match the natural shape of the domain where an emission emitted at a point is detectable.

References

1 E. Aboufadel, T. Armstrong, and E. Smietana. Position coding. *arXiv preprint*, 2007. `arXiv:0706.0869`.

2 R. Berkowitz and S. Kopparty. Robust positioning patterns. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1937–1951, 2016.

3 A. Blanca and A. Godbole. On universal cycles for new classes of combinatorial structures. *SIAM Journal on Discrete Mathematics*, 25(4):1832–1842, 2011. Publisher: SIAM.

4 A. Bruckstein, T. Etzion, R. Giryes, N. Gordon, R. Holt, and D. Shuldiner. Simple and robust binary self-location patterns. *IEEE Transactions on Information Theory*, 58(7):4884–4889, 2012.

5 J. Burns and C. Mitchell. *Coding schemes for two-dimensional position sensing*. Hewlett-Packard Laboratories, Technical Publications Department, 1992.

6 Y. Chee, D. Dao, H. Kiah, S. Ling, and H. Wei. Binary robust positioning patterns with low redundancy and efficient locating algorithms. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2171–2184, 2019.

7 C. S. Chen, P. Baracca, E. de Panafieu, and D. Michalopoulos. A positioning method exploiting ambient IoT deployments in controlled environments, Nokia Patent Application, 2023.

8 Chung Shue Chen, Peter Keevash, Sean Kennedy, Élie de Panafieu, and Adrian Vetta. Robot positioning using torus packing for multisets, 2024. `arXiv:2404.09981`.

9 F. Chung, P. Diaconis, and R. Graham. Universal cycles for combinatorial structures. *Discrete Mathematics*, 110(1-3):43–59, 1992. Publisher: North-Holland.

10 D. Curtis, T. Hines, G. Hurlbert, and T. Moyer. Near-universal cycles for subsets exist. *SIAM Journal on Discrete Mathematics*, 23(3):1441–1449, 2009.

11 M. Dębski and Z. Lonc. Universal cycle packings and coverings for k-subsets of an n-set. *Graphs and Combinatorics*, 32:2323–2337, 2016. Publisher: Springer.

12 N. de Bruijn. Acknowledgement of priority to C. Flye Sainte-Marie on the counting of circular arrangements of 2^n zeros and ones that show each n-letter word exactly once. *EUT report. WSK, Dept. of Mathematics and Computing Science*, 75-WSK-06, 1975.

13 É. de Panafieu. Torus packing for multisets. Software, version 1.0., swhId: `swh:1:dir:2eb591ca09fb8739e2135e17b5f595b7f25ed924` (visited on 2024-06-17). URL: `https://gitlab.com/depanafieuelie/torus-packing-for-multisets`.

14 S. Glock, F. Joos, D. Kühn, and D. Osthus. Euler tours in hypergraphs. *arXiv preprint*, 2020. `arXiv:1808.07720`.

15 N. Hartsfield and G. Ringel. *Pearls in graph theory: A comprehensive introduction*. Courier Corporation, 2013.

16 S.-W. Ho and C. S. Chen. Visible light communication based positioning using color sensor. In *IEEE 8th Optoelectronics Global Conference (OGC)*, pages 1–5, 2023.

17 G. Hurlbert and G. Isaak. On the de Bruijn Torus problem. *Journal of Combinatorial Theory, Series A*, 64(1):50–62, 1993. Publisher: Elsevier.

18 G. Hurlbert and G. Isaak. New constructions for de Bruijn tori. *Designs, Codes and Cryptography*, 6(1):47–56, 1995.

19 G. Hurlbert, T. Johnson, and J. Zahl. On universal cycles for multisets. *Discrete Mathematics*, 309(17):5321–5327, 2009. `doi:10.1016/j.disc.2008.04.050`.

20 G. Hurlbert, C. Mitchell, and K. Paterson. On the existence of de Bruijn tori with two by two windows. *Journal of Combinatorial Theory, Series A*, 76(2):213–230, 1996. Publisher: Elsevier.

21 D. Knuth. *Art of Computer Programming, Combinatorial Algorithms*, volume 4A. Addison-Wesley Professional, 2011.

22 H. Li, C. Zhou, S. Wang, Y. Lu, and X. Xiang. Two-dimensional gold matrix method for encoding two-dimensional optical arbitrary positions. *Optics Express*, 26(10):12742–12754, 2018.

23 J. Luo, L. Fan, and H. Li. Indoor positioning systems based on visible light communication: State of the art. *IEEE Commun. Surveys Tuts.*, 19(4):2871–2893, 2017.

24 J. MacWilliams and N. Sloane. Pseudo-random sequences and arrays. *Proceedings of the IEEE*, 64(12):1715–1729, 1976.

25 D. Makarov and A. Yashunsky. On a construction of easily decodable sub-de Bruijn arrays. *Journal of Applied and Industrial Mathematics*, 13:280–289, 2019. Publisher: Springer.

26 M. Rahman, T. Li, and Y. Wang. Recent advances in indoor localization via visible lights: A survey. *Sensors*, 20(5), 2020.

27 D. Schüsselbauer, A. Schmid, and R. Wimmer. Dothraki: Tracking tangibles atop tabletops through de-Bruijn tori. In *Proceedings of the 15th International Conference on Tangible, Embedded, and Embodied Interaction*, pages 1–10, 2021.

28 F. Sinden. Sliding window codes. *AT&T Bell Labs Technical Memorandum*, 1985.

29 I. Szentandrasi, M. Zachariáš, J. Havel, A. Herout, M. Dubska, and R. Kajan. Uniform marker fields: Camera localization by orientable de Bruijn tori. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 319–320, 2012.

30 Z. Yang, Z. Wang, J. Zhang, C. Huang, and Q. Zhang. Wearables can afford: Light-weight indoor positioning with visible light. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 317–330, 2015.

31 F. Zafari, A. Gkelias, and K. Leung. A survey of indoor localization systems and technologies. *IEEE Commun. Surveys Tuts.*, 2019.

A Software Implementation

We implemented our algorithm constructing the grid colouring as well as the decoding algorithm in Python. The code is available at [13]. An example of grid colouring computed with this code is presented in Figure 4.

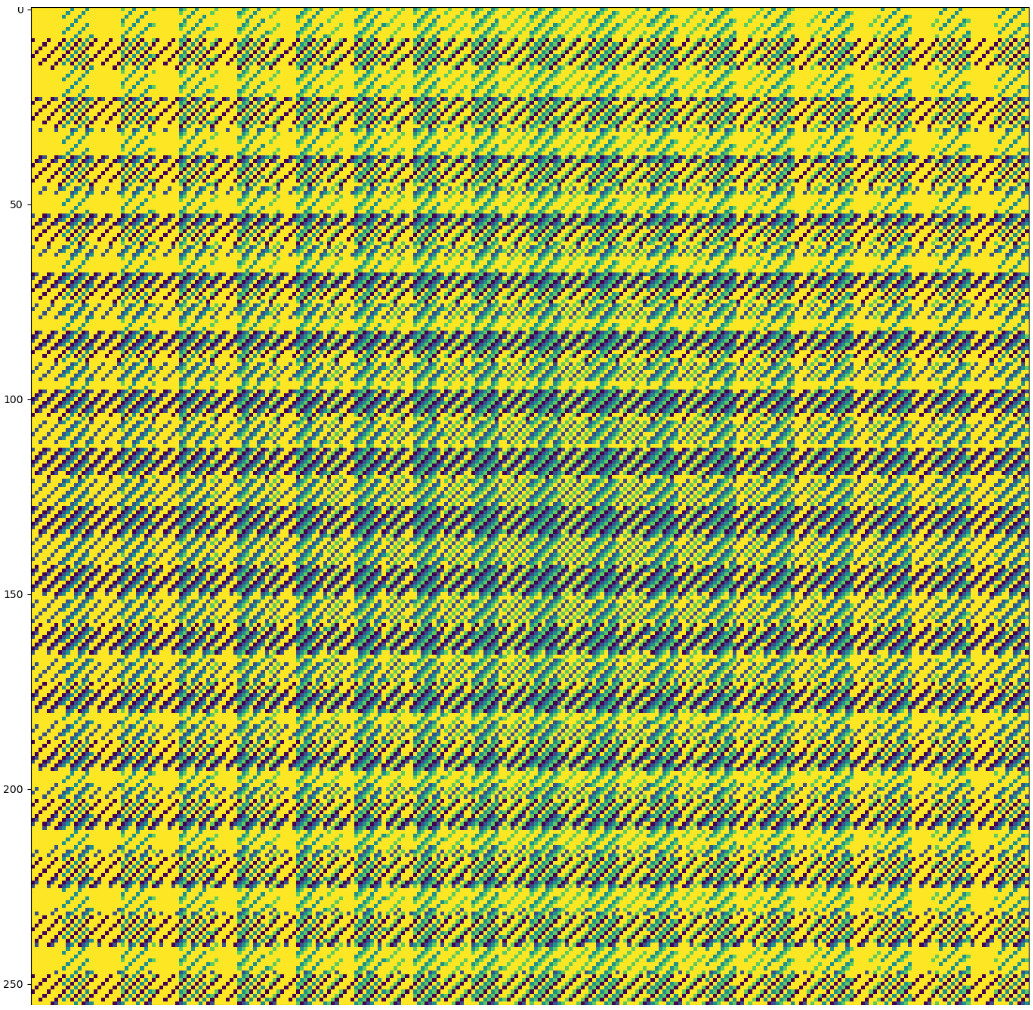

Figure 4 Grid colouring of size 256, window size 8 and number of colours 5. It corresponds to the parameters $d = 2$, $b = 2$ and $t = 1$.