[Draft] Basis Decomposition and Efficient Encoding of Box Pleating Crease Patterns

Brandon Wong
Stanford University
wongb@stanford.edu

October 30, 2025

Abstract

We present a novel method for representing box pleating origami crease patterns as fixed-size binary vectors through decomposition over a basis of fundamental subcrease patterns. We define a superposition operation on crease patterns and construct a linearly independent basis of $2k^2 + 2k - 2$ subcrease patterns that spans all flat-foldable crease patterns on a $k \times k$ grid. Any crease pattern can be uniquely decomposed into binary coefficients over this basis by solving a linear system over the finite field GF(2). In addition to losslessly compressing box pleated crease patterns orders of magnitude more compact than the conventional .FOLD format, this approach reimagines crease patterns from graph-structured objects into points in a well-defined vector space, opening new avenues for computational origami design.

0.1 Introduction

Suppose we want to express a crease pattern as a fixed size vector. This would primarily be useful for optimization or machine learning applications, where it is convenient to represent crease patterns in points in a high-dimensional space rather than as a graph as the conventional crease pattern data structure. We will assume that all vertices lie on a $k \times k$ grid, and creases are either horizontal, vertical, or diagonal at 45 degrees. We will also ignore mountain valley assignment, and assume that all creases are fully folded (i.e., no 3D fold angles).

The naive approach would be to represent the crease pattern as a binary vector of length $4k^2 - 2k$, where each entry corresponds to a possible crease in the $k \times k$ grid, and is 1 if the crease is present and 0 otherwise. Let us call these unit-long (or $\sqrt{2}$-long for diagonal creases) creases *subcreases*, and this vector a *subcrease vector* $\vec{b}$.

We can improve our representation by taking inspiration from image compression by frequency decomposition. Instead of representing the crease pattern in terms of individual subcreases, we can represent it as a "superposition" of basis subcrease vectors called *subcrease patterns*. Let $\vec{b}^{(1)}, \vec{b}^{(2)} \in \{0,1\}^m$ be the binary vectors representing two crease patterns, where 1 indicates the presence of a crease and 0 its absence. Then, we define the *superposition* of the two patterns as

$$\vec{b}^{\text{super}} = (\vec{b}^{(1)} + \vec{b}^{(2)}) \bmod 2.$$

This is an elementwise XOR operation: take the union of creases from each pattern, but cancel out creases that overlap. This makes intuitive sense if you think of a crease as a reflection or 180-degree rotation.

0.2 Subcrease pattern basis

Each subcrease pattern can be represented as a subcrease vector. We then want to find a basis of subcrease patterns that span the space of all flat foldable crease patterns on a $k \times k$ grid and are linearly independent under our defined superposition. We can put these vectors as columns in a matrix A_k. Then, any flat foldable crease pattern on a $k \times k$ grid can be represented as a vector of binary coefficients $\vec{c}$ that can be used to reconstruct the crease pattern by left multiplying by A_k and taking mod 2:

$$A_k \vec{c} \mod 2 = \vec{b} \tag{1}$$

Let $m = 4k^2 - 2k$ be the number of subcreases in a $k \times k$ grid. Then, A_k is an $m \times n$ matrix, where n is the number of basis subcrease patterns. It turns out that $n = 2k^2 + 2k - 2$ for a linearly independent basis of subcrease patterns. This includes all edge-to-edge horizontal, vertical, and diagonal creases, and for each internal vertex: a "V" subcrease patterns with two diagonal creases radiating out, and an "L" subcrease pattern with one horizontal and one vertical crease radiating out. The basis A_k only needs to be precomputed once per value of k and can then be reused for any crease pattern on that grid size. An example basis of subcrease patterns for $k = 3$ is shown in Fig. 1, and the matrix form A_3 is shown in Eq. (2). Note that the indexing scheme is arbitrary.

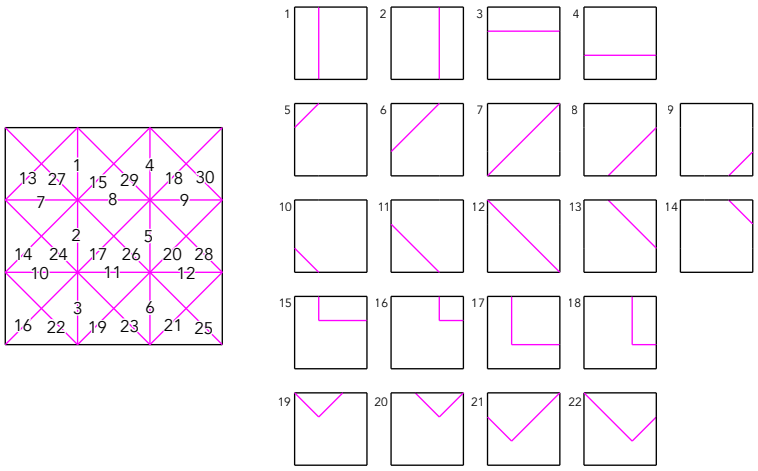


Figure 1: Basis subcrease patterns for $k = 3$. There are $2(3^2) + 2(3) - 2 = 22$ basis subcrease patterns, each represented as a binary subcrease vector of length $4(3^2) - 2(3) = 30$.

0.3 Decomposition

Because A_k has full column rank, we can decompose any flat foldable crease pattern b on a kxk grid into a unique vector of coefficients c by solving the linear system $A_k c = b$ over the finite field $\mathbb{GF}(2)$. This can be done efficiently using Gaussian elimination. The resulting coefficient vector c then represents the crease pattern as a superposition of the basis subcrease patterns. An example of a crease pattern in its decomposed and recontructed form is shown in Fig. 2.

0.4 Conclusion

There are a lot of reasons why representing crease patterns as a vector of coefficients over a fixed basis of subcrease patterns is useful:

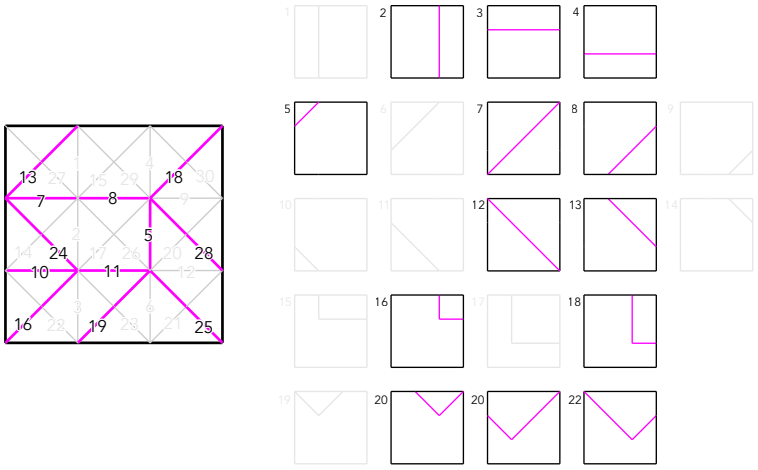


Figure 2: An example of a crease pattern for $k = 3$, along with its decomposition into basis subcrease patterns. Refer to Eq. (2) to see how the subcrease vector, the subcrease pattern coefficient vector, and the basis matrix A_3 relate to each other.

0.4.1 Fixed-size representation

This representation satisfies our initial motivation by encoding any flat-foldable crease pattern on a $k \times k$ grid as a fixed-size vector of length $2k^2 + 2k - 2$, regardless of the crease pattern's complexity. This fundamentally reframes how we conceptualize crease patterns: rather than planar graphs with variable structure, they become points in a well-defined vector space. This perspective enables several new approaches. First, optimization problems transition from discrete graph optimization to continuous function optimization over a vector space, making gradient-based methods applicable. Second, the vector space structure naturally supports geometric operations such as computing distances between crease patterns (via norms like ℓ^2 or Hamming distance) and interpolating between designs to explore intermediate forms. Third, the representation's dimensionality—approximately half that of the naive subcrease vector representation—significantly reduces the search space for optimization and learning algorithms without sacrificing expressiveness compared to the naive representation.

0.4.2 Flat foldability

It's not quite the case that all crease patterns in the space of this representation will be flat foldable. However, this representation guarantees that all internal vertices are even degree, and even if they violate Kawasaki foldability, they can only do so with an angle deficit of 90 degrees (rather than 45 or 135 degrees).

0.4.3 Storage efficiency

For a crease pattern on a grid size k, the representation can be stored in exactly $2k^2 + 2k - 2$ bits. For any crease pattern on a 16x16 grid, that's around 70 bytes, which is less than the size of a couple of sentences in a text file. In contrast, the typical .FOLD format is generally orders of magnitude larger, because each vertex needs to be stored as a pair of floats (8 bytes per vertex) and each edge needs to be stored as a pair of vertex indices (8 bytes per edge). This compression is lossless but comes at the price of constraining the crease pattern to a strict box pleating format.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	**1**	0	0	0	0	0
2	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
3	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	**1**	0	0	0	0
5	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0
6	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
9	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0
10	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
12	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0
13	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
16	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
18	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0
19	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
21	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	**1**	0
25	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	**1**
27	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	**1**	0	0	**1**
28	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	**1**	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0

$$\begin{bmatrix} 0 \\ \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \\ \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ \mathbf{1} \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

Equation 3: The expanded form of the equation $(A_k\vec{c}) \mod 2 = \vec{b}$ for $k = 3$ and $\vec{c}, \vec{b}$ from the crease pattern shown in Fig. 2. (A_3 is a matrix looks like a table because indices have been provided for the viewer's convenience). Each column corresponds to a basis subcrease pattern, and each row corresponds to a subcrease in the 3×3 grid. Red entries highlight where a crease is present in the basis subcrease pattern. The matrix A_3 can be precomputed once and reused for any crease pattern on a 3×3 grid.