Prospecting for pluripotency in metamaterial design

Mastrigt, R. van; Dijkstra, M.; Hecke, M.L. van; Coulais, C.

Editors' Suggestion

Prospecting for pluripotency in metamaterial design

Ryan van Mastrigt [1,2,3,*] Marjolein Dijkstra [4] Martin van Hecke [2,5] and Corentin Coulais [1]

[1] *Institute of Physics, Universiteit van Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands*

[2] *AMOLF, Science Park 104, 1098 XG Amsterdam, The Netherlands*

[3] *Gulliver UMR CNRS 7083, ESPCI Paris, PSL University, 10 rue Vauquelin, 75005 Paris, France*

[4] *Soft Condensed Matter & Biophysics, Debye Institute for Nanomaterials Science, Department of Physics, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands*

[5] *Huygens-Kamerling Onnes Lab, Universiteit Leiden, Postbus 9504, 2300 RA Leiden, The Netherlands*

(Received 9 December 2024; accepted 8 May 2025; published 24 June 2025)

From self-assembly and protein folding to combinatorial metamaterials, a key challenge in material design is finding the right combination of interacting building blocks that yield targeted properties. Such structures are fiendishly difficult to find—not only are they rare, but often the design space is so rough that gradients are useless and direct optimization is hopeless. Here, we design ultrarare combinatorial metamaterials, capable of multiple desired deformations, by introducing a twofold strategy that avoids the drawbacks of direct optimization. We first combine convolutional neural networks with genetic algorithms to prospect for metamaterial designs with a potential for high performance; in our case, these metamaterials have a high number of spatially extended modes—they are pluripotent. Second, we exploit this library of pluripotent designs to generate metamaterials with multiple target deformations, which we finally refine by strategically placing defects. Our multishape metamaterials would be impossible to design through trial-and-error or standard optimization. Instead, our data-driven approach is systematic and ideally suited to tackling the large and intractable combinatorial problems that are pervasive in material science.

DOI: 10.1103/PhysRevResearch.7.023299

I. INTRODUCTION

Designing new materials requires efficient generation and evaluation of candidate designs. For varying continuous parameters, such as the density of a material, the design approach is well established: designs iterate through gradients of an (approximate) objective function [1–19]. For designs that require combining building blocks, no gradients exist. Such combinatorial design problems are ubiquitous in science and can be found, e.g., in protein design [20–22], combinatorial chemistry [23–25], DNA architectures [26,27], and metamaterial design [28–31]. Without a gradient, design approaches typically rely on the structure of the design space, e.g., by metaheuristics [32–34] or by machine learning search strategies [35]. However, for problems with many local optima that are uncorrelated to further improvements to the objective function, i.e., that have a rugged design space, such strategies often fail [36].

Such rugged design spaces are particularly prevalent in mechanical metamaterials [37]—synthetic materials that leverage geometric effects to achieve novel mechanical properties [38–44]—because designing their topology requires carefully arranging deformable building blocks: an inherently combinatorial challenge. We focus here on oligomodal metamaterials. These exhibit a small number of spatially textured soft modes to achieve exceptional functionalities, such as selective mechanical responses [45–50], nonlocal resonances [51], multi-shape folding [52–59], and sequential energy-absorption [60]. To design such metamaterials, combinatorial strategies based on a small number of building blocks are ideal as one can consider the deformations of individual building blocks [Figs. 1(a) and 1(b)] [29,47]. However, desired combinations of building blocks are like a jigsaw puzzle: rotating a single puzzle piece can render the combination incompatible [61]. This problem is emblematic of combinatorial design: the design space is extremely rugged, explicit design rules are unavailable, and evaluating the modes for each design is computationally heavy. Consequently, the systematic design of combinatorial metamaterials remains an open problem.

Here, we introduce a two-step data-driven approach to design combinatorial metamaterials. To find rare designs with desired modes, we initially search for designs with high *pluripotency*: a statistical measure of a design's ability to deform (softly) over a diverse set of randomized target modes. Crucially, pluripotency provides structure to the design space. The total design approach consists of two steps: (i) prospecting for a library of high pluripotency candidate designs using a genetic algorithm (GA) guided by a convolutional neural network (CNN) [Fig. 1(c)] and (ii) extracting and refining candidate designs to find a design with targeted modes [Fig. 1(d)]. We leverage pluripotency not as a goal but as a tool to tackle rugged combinatorial design, enabling the design of combinatorial metamaterials with multiple unique, targeted deformations that are otherwise difficult to design.

*Contact author: ryan.van-mastrigt@espci.psl.eu

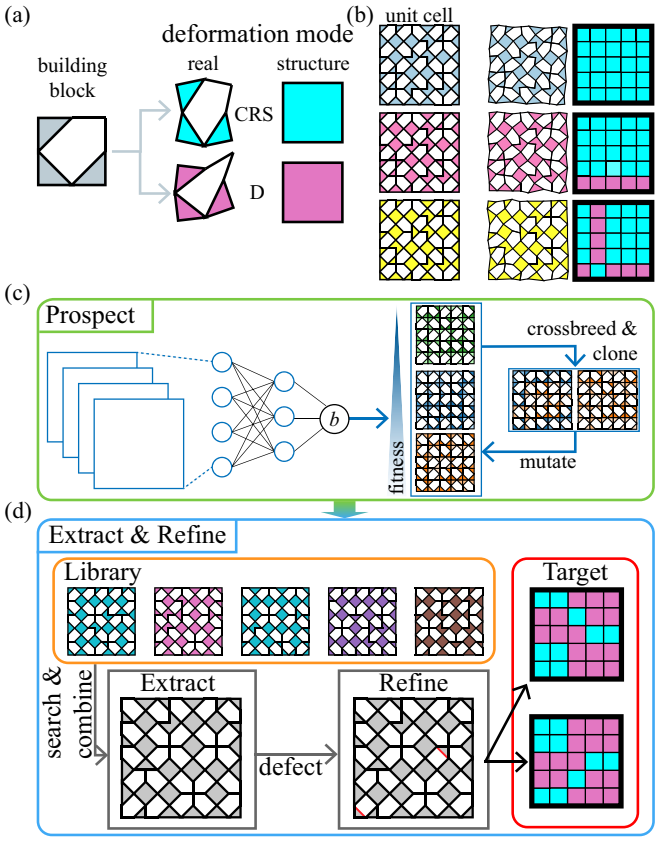

FIG. 1. Design approach for combinatorial metamaterials. (a) The two-dimensional building block (left, gray) can be tiled in four orientations and features two distinct zero modes (middle), which we label CRS (top, cyan) and D (bottom, pink). We visualize the mode structure of a deformed building block as a cyan or pink square (right). (b) The building blocks of (a) combine into larger 5×5 unit cells (left) that feature intensive zero modes (middle). The mode structure of such intensive modes is shown on the right. The three unit cells (blue, pink, yellow) differ only by a single building block from their neighbors, yet support an increasing number of modes: each unit cell supports the zero modes of their top neighbors in addition to the zero mode directly to the right of the unit cell. For example, all possible unit cells support the top zero mode, which we refer to as the counter-rotating squares (CRS) mode. (c) Step (i) of our design approach uses a convolutional neural network (CNN) that predicts the number of intensive modes b to guide a genetic search algorithm (GA) to efficiently generate high b (high pluripotency) designs. (d) In step (ii), the generated high-pluripotency designs are added to a library. We then search and combine designs from this library to find a design that closely matches a set of target deformations (red): this process is called extraction. Finally, our extracted designs often require further refinement which we implement by strategically introducing defects. The final refined design features the desired target deformation modes while minimizing undesired superfluous modes.

Our two-step approach is eminently suited to design oligomodal metamaterials with multiple target modes without requiring design rules or continuous optimization. Previously, combinatorial metamaterial design was limited to a single target mode and known rules [29,56,62] or incorporated into continuous strategies [63–68]. Specifically, we consider a family of mechanical metamaterials with multiple soft modes [47] [Figs. 1(a) and 1(b)] and demonstrate our approach by designing large metamaterials with two desired textured modes that resemble a smiley and frowny face, or the letters A and U, while minimizing superfluous modes. We stress that designing such a material would be impossible through conventional approaches. Beyond metamaterial design, we anticipate that our approach will be applicable to a wide range of combinatorial problems characterized by multiple, independent sets of constraints to satisfy. Such problems can readily be found in fields such as protein folding [22,69], self-assembly [70,71], computer graphics [72,73], and molecular design [74].

II. MULTIMODAL METAMATERIAL AND DESIGN STRATEGY

To test our design approach, we consider the spatial structure of zero modes—infinitesimal deformations that do not stretch any bonds to first order—in a multimodal combinatorial metamaterial [Figs. 1(a) and 1(b)] [47,49,61], see Appendix A for more details. We design this metamaterial's unit cells by tiling building blocks constructed from rigid bars and hinges on a square lattice. Note that our building block is essentially a five-bar linkage with two deformation modes and four distinct orientations. The orientation of each building block in a unit cell determines both the number and spatial structure of the zero modes, leading to a rough design space where changing a single building block can completely alter the structures of the zero modes. By controlling the structure and enumeration of zero modes in aperiodic multimode configurations, we can design complex mechanical functionalities that can be actuated as needed, with potential applications in programmable materials, soft robotics and computing *in materia* [47,49,51,60]. To design such spatially textured zero modes, primarily discrete rule-based techniques have been developed [29,62,75]. However, the problem is that if such rules cannot be derived, computational approaches appear hopeless because of the sheer size of the design space and high sensitivity of designs to changing building blocks. Additionally, these techniques are limited to combinatorial metamaterials that support a single spatially textured zero mode. The task of designing oligomodal metamaterials that feature multiple target modes is significantly harder and systematic techniques are lacking.

As a first step, we aim to design pluripotent metamaterials that feature many spatially extended modes. Such modes span the entire structure, allowing for a rich diversity of selectable deformations while limiting the number of deformational degrees of freedom. For our metamaterials, the number of zero modes, $N_M(n)$, (see Appendix A 2 for how we calculate N_M) for an $n \times n$ tiling of unit cells (each containing $k \times k$ building blocks) grows with n as

$$N_M = an + b, \tag{1}$$

where a and b are integers that correspond to the number of modes in the unit cell that grow linearly or that remain constant upon repeated tiling of the unit cell. We refer to modes that count towards a as additive modes and modes that count towards b as intensive modes. Additive modes are necessarily spatially localized, while intensive modes have the

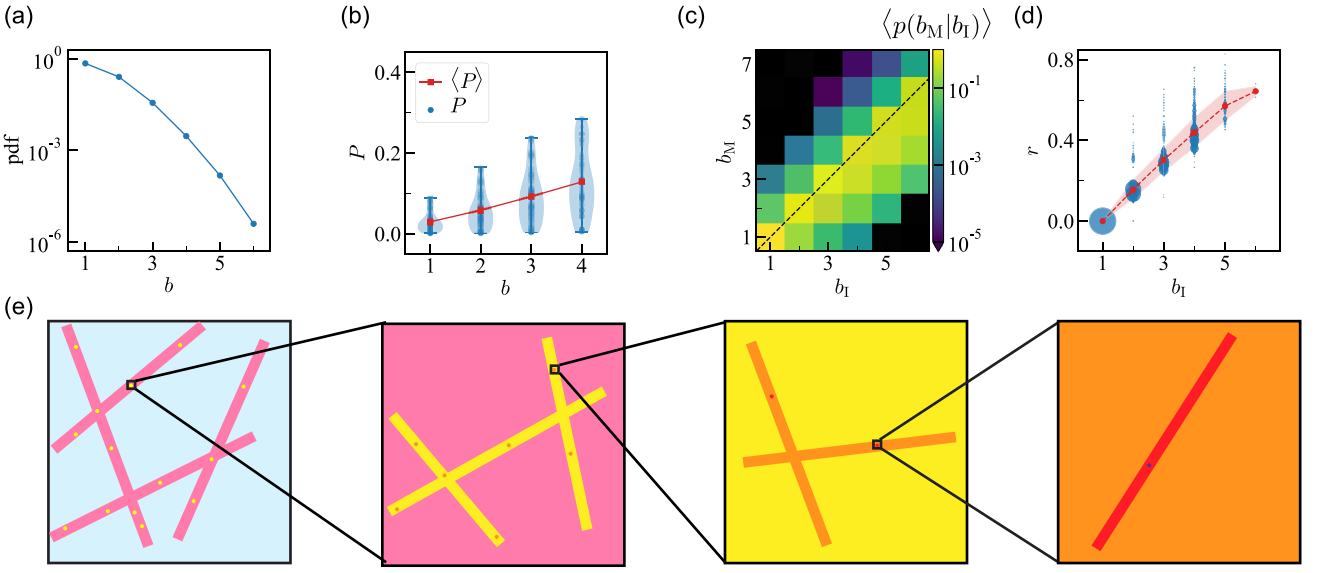

FIG. 2. Statistical characterization of the design space. (a) Probability density function (pdf) for the number of intensive modes b obtained through Monte Carlo sampling of 5×5 unit cells. (b) An increasing number of intensive modes b correlates to a higher average capacity [Eq. (2)] or pluripotency P per design (blue circles). The pluripotency averaged over random designs (red squares) increases with b. The blue shaded area represents a violin plot to visualize the distribution of P values. (c) The average probability $\langle p(b_M|b_I)\rangle$ of finding a mutated unit cell with b_M intensive modes under point mutations for an initial unit cell with b_I intensive modes. To highlight the low probability of transitioning to a higher b, a logarithmic colormap (colorbar) is used, with zero (measured) probability denoted in black. (d) The average ratio r (red dashed line, shaded area indicates the standard deviation) of mutated unit cells with $b_M \leqslant b_I$ averaged over random unit cells with b_I intensive modes increases with b_I. The local r varies for individual unit cells (blue circles, where the size indicates the number of unit cells with the same r) and appears multimodal in structure. This is likely a consequence of the different types of intensive modes: edge and global (see Appendix A 5). (e) Conceptual configuration space of a discrete combinatorial multimodal metamaterial problem. Areas of the same color represent designs with the same b, where blue, pink, yellow, orange, and red correspond to areas with progressively increasing b. Random point-mutations of the design—rotating a single building block—generally lower the number of intensive modes b, most designs have a low b and we have few examples of rare designs with higher values of b. Here, satisfying a hierarchy of conditions leads to increasingly low-dimensional subspaces where b has larger values (left to right).

potential to span the entire metamaterial [49]. We previously investigated the detection of rare unit cells with $a \geqslant 1$ using CNNs, showed that these unit cells contain specific additive 'strip' modes, and investigated their design and properties [49,60,61]. In contrast, here we focus on rare pluripotent designs with spatially extended zero modes, which requires many intensive modes (large b) and no strip modes ($a = 0$). For all designs, $b \geqslant 1$ due to the presence of the well-known counter-rotating squares (CRS) mode [37,38,47,76–81] [Fig. 2(a)], shown in the top row of Fig. 1(b). Yet, there are no design rules for realizing larger values of b, illustrating the need for the development of an effective search algorithm.

We use the deformation in each building block to characterize the spatial structure of a zero mode M. We follow a framework devised in earlier work [49], see Appendix A 3 for a brief description of the framework. In particular, we decompose the deformations of the constituent building blocks m_i into the trivial CRS deformation m_{CRS} and the nontrivial deformation $m_D(c_i)$, where c_i denotes the orientation of building block i: $m_i = \alpha_i\, m_{\mathrm{CRS}} + \beta_i\, m_D(c_i)$, see Fig. 1(a) for the deformed building blocks. We label building block deformations with $\beta = 0$ as CRS blocks and those with $\beta \neq 0$ as D blocks [Fig. 1(a)]. Note that CRS blocks leave the angle of the corner at the vertex of the outer square unchanged while D blocks do not—the trivial CRS mode is composed solely of CRS blocks [Fig. 1(b)]. To distinguish between areas in the metamaterial that deform as the trivial CRS mode and those that do not, we characterize the structure of zero modes by the spatial distribution of CRS and D blocks, see Appendix A 4 for limitations to the types of structures this metamaterial can support. We define a set of target modes $\{\hat{M}\}$ by its CRS and D blocks, i.e., the target mode's elements $\hat{m}_i$ are simply labeled CRS or D [Fig. 1(d)]. Unlike our earlier work, we do not aim to derive design rules. Instead, we use our two-step approach to construct a design that supports the target modes.

To start our approach, we must quantify the pluripotency of a given $k \times k$ design, which involves considering both a set of zero modes of the material $\{M\}$ and a set of targeted zero modes $\{\hat{M}\}$. We define the similarity between modes M and $\hat{M}$ as the fraction of matching CRS and D blocks:

$$S(M,\hat{M}) = \frac{1}{k^2}\sum_{i=1}^{k^2}\begin{cases}\delta(\beta_i,0), & \text{if } \hat{m}_i = \mathrm{CRS}\\ 1-\delta(\beta_i,0), & \text{if } \hat{m}_i = \mathrm{D}\end{cases}, \qquad (2)$$

where δ represents the Kronecker delta and $\hat{m}_i$ is the element of target mode $\hat{M}$ at building block i. The sum runs over all $k \times k$ building blocks. We then define the capacity of a design as the maximum similarity between the target mode and linear combinations of the zero modes $\{M^j\}$ of the design:

$$C(\{M^j\},\hat{M}) = \max_{\mathbf{w}}\left[S\left(\sum_j w_j M^j,\hat{M}\right)\right] - \hat{N}_{\mathrm{CRS}}, \qquad (3)$$

where w_j are real-valued weights, j labels the weights and modes, and $\hat{N}_{\mathrm{CRS}}$ is the fraction of CRS blocks in the target mode. We include $\hat{N}_{\mathrm{CRS}}$ to set the minimal capacity to 0, as a trivial linear combination of $\mathbf{w} = \mathbf{0}$ is equal to $\hat{N}_{\mathrm{CRS}}$. We calculate the capacity using constraint programming, which aims to find the set of weights that maximize S (see Appendix A 6 for more details and Appendix F for pseudocode). We finally define the pluripotency P of a design as the capacity C averaged over a randomized set of generated target modes $\{\hat{M}\}$:

$$P(\{M\}, \{\hat{M}^i\}) = \frac{1}{N_{\mathrm{T}}} \sum_i C(\{M\}, \hat{M}^i), \tag{4}$$

where N_{T} is the number of target modes in the set and i labels each target mode. See Appendix A 7 for how we generate the target modes.

Of course, the capacity is maximal for extremely floppy designs, i.e., designs with an extremely large number of deformational degrees of freedom, but their excess zero modes hinder their functionality [50]. Ultimately, we aim to construct metamaterials which maximize C without any excess zero modes, and our oligomodal combinatorial metamaterials are ideally suited to do so [47].

We now show that designs with a larger number of intensive modes b result in higher pluripotencies. We first generate a random set of 100 target modes (see Appendix A 7 for all possible target modes). We then generate 10^6 random designs, calculate b for each, and then calculate their pluripotency P as described before. We find that the pluripotency steadily increases with the number of intensive modes b of the design [Fig. 2(b)]. Thus we take b as a proxy for pluripotency P, because b is computationally cheaper to calculate than P.

III. HIGH PLURIPOTENCY DESIGNS

We now prospect for high pluripotency by searching for designs with values of b that are larger than can be obtained by random sampling. While we have indications for their existence, we lack examples of such designs.

We first explore the design space of 5×5 unit cells, as this size offers a good balance between spatial complexity, the size of the design space, and the rarity of the number of intensive modes b. We note that designs with large values of b are increasingly rare [Fig. 2(a)], yet those are interesting because they tend to be pluripotent [Fig. 2(b)].

Crucially, there is structure within the design space, which we explore by characterizing the changes in the number of intensive modes b under point mutations of the design [Fig. 2(c)]. We define point mutations as a random change of orientation of one of the building blocks. Comparing the number of intensive modes of the initial design, b_{I}, to the number of the mutated design, b_{M} we find that the most likely scenario is that the number remains unchanged, i.e., $b_{\mathrm{M}} = b_{\mathrm{I}}$, showing that designs with the same value of b are interconnected. The next most likely scenario is that the number decreases by one, i.e., $b_{\mathrm{M}} = b_{\mathrm{I}} - 1$, showing that designs with b zero modes are typically surrounded in design space by designs with $b - 1$ zero modes. The rarest yet desired scenario is that the number increases, i.e., $b_{\mathrm{M}} > b_{\mathrm{I}}$.

TABLE I. Confusion matrix for the test set of the CNN with the lowest validation loss.

		predicted b_{CNN}				
		1	2	3	4	5
actual b	1	105 767	47	0	0	0
	2	418	37 877	38	0	0
	3	1	242	5101	16	0
	4	0	0	66	383	3
	5	0	0	1	5	17

To gain further insight, we explored the ratio r of neighboring designs (designs that differ by a single building block orientation) that have $b_{\mathrm{M}} < b_{\mathrm{I}}$, and found that this ratio increases with b_{I}, showing that subspaces with large b have increasingly small dimensionality [Fig. 2(d)]. This means that the subspaces of the design space with constant b become progressively sparse and low-dimensional with increasing b. We describe this hierarchical structure intuitively as "needles-within-needles-within-needles in a haystack" [Fig. 2(e)], which stands in stark contrast to the less structured, jagged structure of the direct objective function in design space.

To find rare high b designs, we first turn our attention to a computationally effective method to estimate b. A straightforward calculation of b requires determining the number of zero modes as function of the number of unit cells n, which is computationally demanding [82,83]. Inspired by the earlier success of convolutional neural networks (CNNs) in classifying designs with $a \geqslant 1$, we now set out to use CNNs to estimate b [61]. In contrast to our previous work, the underlying structure in design space is hierarchical in nature [Fig. 2(e)]. It is unclear, *a priori*, whether the CNN can effectively capture this hierarchical structure, especially as samples with increasing b become increasingly sparse in the training set. We train our CNN using training data obtained by Monte Carlo sampling of the design space for 5×5 unit cells and calculating b from the values of $N_M(n)$ for $n \in \{2, 3, 4\}$ (see Appendix B 3 for details on the training procedure). We test our trained network on a new set of designs that were not used to train the network. We report a low root mean squared error (RMSE) of RMSE ≈ 0.075 over this test set. Crucially, despite the sparsity of designs with a high number of intensive modes b in the training set, we find that the CNN remains accurate for high b in the test set (Table I). In particular, the output of the CNN, b_{CNN}, is consistently close to the actual number of intensive modes b, albeit with a slight underestimation bias. Crucially, our CNN is two orders of magnitude faster and is readily parallelizable [61]. Thus the CNN is able to quickly and accurately detect the needles for low b; whether it can identify ultrarare designs with large b remains an open question at this point.

Finally, we now employ a combination of a genetic algorithm (GA) with the trained CNN to iteratively progress towards designs with a high number of intensive modes b (see Appendix C for details on our GA implementation). GAs are a standard tool for searching combinatorial spaces and have been successfully applied to design metamaterials [12,84,85].

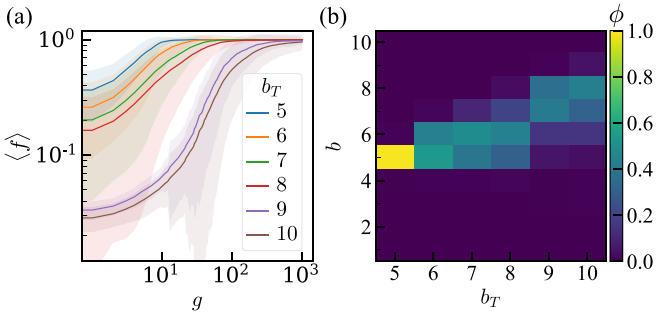

FIG. 3. Extrapolation of the trained CNN. (a) The average (solid line) and standard deviation (shaded area) of the fitness f [Eq. (5)] of the fittest design per generation g over two hundred GA runs for each target number of intensive modes b_T (legend). (a) Fraction $\phi(b)$ of the fittest designs with b intensive modes for a given target b_T across two hundred GA runs per b_T.

The primary computational bottleneck in GAs is the evaluation of the desired property. To overcome this bottleneck, we use our trained CNNs. Without our CNNs, the computation time of evaluation increases by a factor 390 and exploration of the design space is unfeasible. However, finding designs with a high b requires the CNNs to accurately identify designs with b values beyond those represented in the training set.

Specifically, we use the GA to find designs with a target number of intensive modes b_T by maximizing the fitness

$$f = \frac{1}{1 + (b_{\mathrm{CNN}} - b_T)^2}. \tag{5}$$

This fitness is iteratively maximized over a group of designs by combining promising designs and changing orientations of randomly chosen building blocks. Each iteration is termed a generation g (see Appendix C for a detailed description). Surprisingly, starting from a Monte Carlo sampled set of unit cell designs, the GA consistently reaches its maximum fitness relatively quickly, even when b_T exceeds the range of the training data, i.e., $b_T > 6$ [Fig. 3(a)]. Even though the CNN overestimates the number of intensive modes, a significant fraction of these designs indeed features $b > 6$ [Fig. 3(b)].

Hence, our combination of GA and CNN is able to identify extremely rare designs with high pluripotency (e.g., up to $b = 9$). We estimate that such designs represent only a fraction on the order of 10^{-8} of the total design space [86]; yet combining GA and CNNs allows us to find these within minutes on a desktop computer. We believe that the hierarchical 'needles-within-needles' structure of the design space is crucial. First, it allows the GA, by a combination of local and nonlocal exploration, to incrementally increase b (see Appendix C 1 for how our GA explores design space, and Appendix C 3 for a quantitative comparison to local search algorithms); second, we believe that this underlying structure allows the CNN to extrapolate beyond its training data. Hence, we have carried out the first step of our design strategy, by optimizing a general quantity—pluripotency—that has structure in the design space, rather than directly optimizing an essentially random specific objective function.

IV. DESIGNING FOR TARGET DEFORMATIONS

We now transform our candidate designs, which have a high pluripotency based on their capacity for random targets, into a specific design with a high capacity for specific targets $\{\hat{M}\}$. This is step (ii) of our design strategy, which itself proceeds in two substeps, which we refer to as extract and refine.

In the extracting step, we first generate a library of 1000 pluripotent "base" designs and use these to generate a much larger space of designs [Fig. 4(a)]. This enables us to identify the combination of modes in the library that most effectively matches the desired mode structures. We set the target for the number of intensive modes for the base designs as $b_T = 7$, and find that the distribution of their actual values of b is centered around $b = 6$ [Fig. 4(b)]. Next, we compute all corresponding zero modes, remove the trivial CRS mode and most edge modes [modes containing single-block-wide strip(s) of D blocks located at the edge of the metamaterial in a background of CRS blocks, see Appendix D for how we remove such modes), thus obtaining the spatially extended zero modes of each design [Fig. 4(a)].

To expand our search space, we use these computed modes to systematically generate new designs by identifying combinations of modes that can be supported by the new design. Specifically, we generate a very large number of high pluripotency designs ("clique designs") as follows. We define two modes as compatible when they have no D blocks with different orientations at the same site [Fig. 4(c)]. We construct a graph where the nodes represent the modes and the edges indicate pairs of compatible modes. We then determine the maximal cliques—sets of nodes such that every two constituent nodes are adjacent (clique) that cannot be extended by adding additional nodes (maximal)—of this graph using the Bron-Kerbosch algorithm [88]. Cliques correspond to sets of compatible zero modes [Fig. 4(c)]. Crucially, while each base design corresponds to a clique, the large number of additional maximal cliques stem from multiple base designs. For each of these, we can rationally combine the relevant base designs to create a new design (the clique design) that precisely generates the modes in the additional clique (see Appendix D for how we combine designs) [Fig. 4(c)–4(d)].

These additional designs significantly expand the search space: the number of maximal cliques exceeds the number of initial designs by a factor 70, and many of these correspond to designs with a high clique size and thus a corresponding high pluripotency [Fig. 4(e)]. Then, to extract a design we select the maximal clique with the highest capacity C [Eq. (3)] with respect to the target modes $\{\hat{M}\}$, and consolidate the constituent designs into a new, highest-capacity, single candidate design that supports all modes in the clique [Fig. 4(d)].

To test the capabilities of this large set of designs, we define 100 random targets and study the success rate of finding a design that contains a set of modes that perfectly match a target ($C = 1$). For single randomized target deformations we successfully find a design in all cases; for two randomized target deformations, we find designs in 82 cases. Hence, extending the design space with maximal cliques allows us to find designs that satisfy multiple target deformations with a high probability.

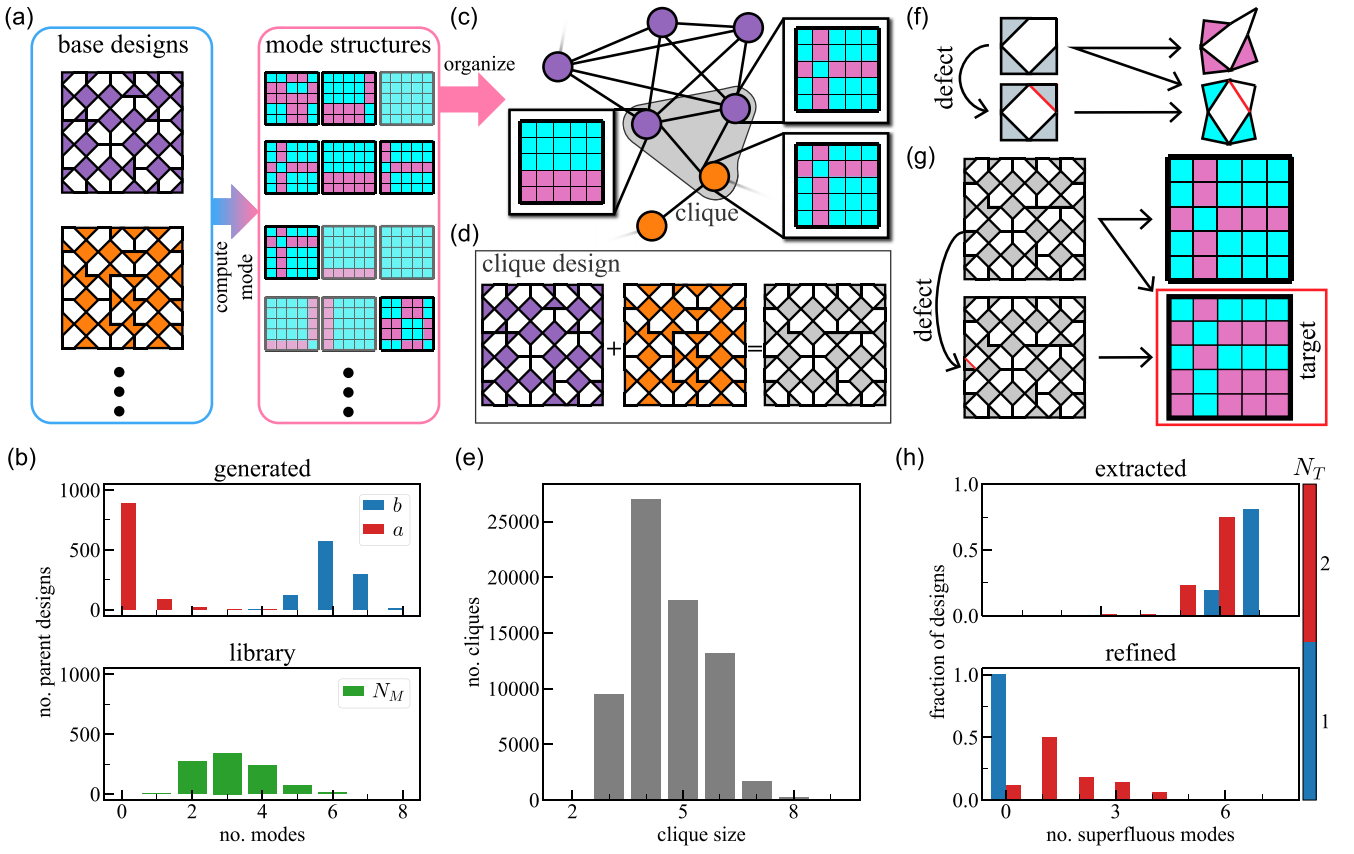

FIG. 4. Extracting and refining high pluripotency designs. (a) We use 1000 GA-generated designs (left), calculate their mode structures (right), and discard the common CRS mode (all blue) and simple edge modes (single-block-wide strip(s) of D blocks located at the edge), as indicated by the grayed-out mode structures. The remaining modes are organized in a graph. (b) Top: histogram of the true number of intensive modes b (blue left bars) and additive modes a (red right bars) for 1000 GA-generated base designs with a target of $b_T = 7$. Bottom: histogram of the number of modes N_M (green bars) after discarding the common CRS mode and edge modes (see Appendix D) per base design in the library. The total number of modes is significantly reduced after this discarding. (c) Individual modes from distinct base designs (indicated by color) are represented as nodes in a graph. Nodes are connected by edges if the corresponding modes are compatible. Each maximal clique (fully connected nodes enclosed in gray area) corresponds to a new clique design [see (d)] that supports all constituent modes. (d) The clique in (c) comprises modes from two base designs (purple and orange) which we combine into a new clique design (gray) that supports all three modes in the clique. (e) Histogram of maximal clique sizes (number of nodes) in the library. We note that a larger clique size corresponds to a higher number of modes and thus a higher (average) pluripotency P [Fig. 2(b)]. (f) Introducing a defect (extra rigid bar, red) to a building block (bottom left) prohibits the D mode (top right, pink) while retaining the CRS mode (bottom right, cyan). Arrows indicate supported modes. (g) Strategic placement of a defect (red) prohibits an undesired mode structure (top right) while retaining the desired target mode structure (bottom right). (h) Normalized histograms of the number of superfluous modes [87] for 100 designs found in our library with the highest cumulative capacity [Eq. (3)] for N_T (colorbar) randomized target deformations before (top) and after (bottom) strategically adding defects (refining). We neglect the trivial global CRS mode, as we are free to remove this mode by adding a single extra horizontal or vertical rigid bar.

Second, we refine. We note that most of our designs originate from large cliques which feature many modes, most of which are superfluous with respect to the target modes. Crucially, removing undesired modes is much easier than adding desired modes: a mode requires a careful selection of building blocks to deform compatibly, while changing a single building block can be sufficient to prohibit the mode. We thus eliminate superfluous modes by introducing point defects in our design [Figs. 4(f) and 4(g)]. Specifically, by adding an extra diagonal bar, each building block can be prevented from deforming in the D mode and can only deform as a CRS block [Fig. 4(f)]. We iterate this procedure until we can no longer transform D blocks to CRS blocks, or until only the desired modes are left. This approach drastically reduces the number of superfluous modes [Fig. 4(h)].

Together, this two-step strategy allows us to obtain targeted pluripotent designs with few superfluous modes: For a single target deformation mode, we are able to completely eliminate any superfluous modes [87]. For two target deformations, most refined designs feature a single superfluous mode, and the number of designs with a higher number of intensive modes rapidly tapers off. Thus we are able to find extremely rare designs that exhibit target deformations while minimizing the number of superfluous deformations.

V. SCALING UP

We now show how we can extend our approach to obtain larger designs by stitching together 5×5 designs. Specifically, we showcase our design approach by example: we

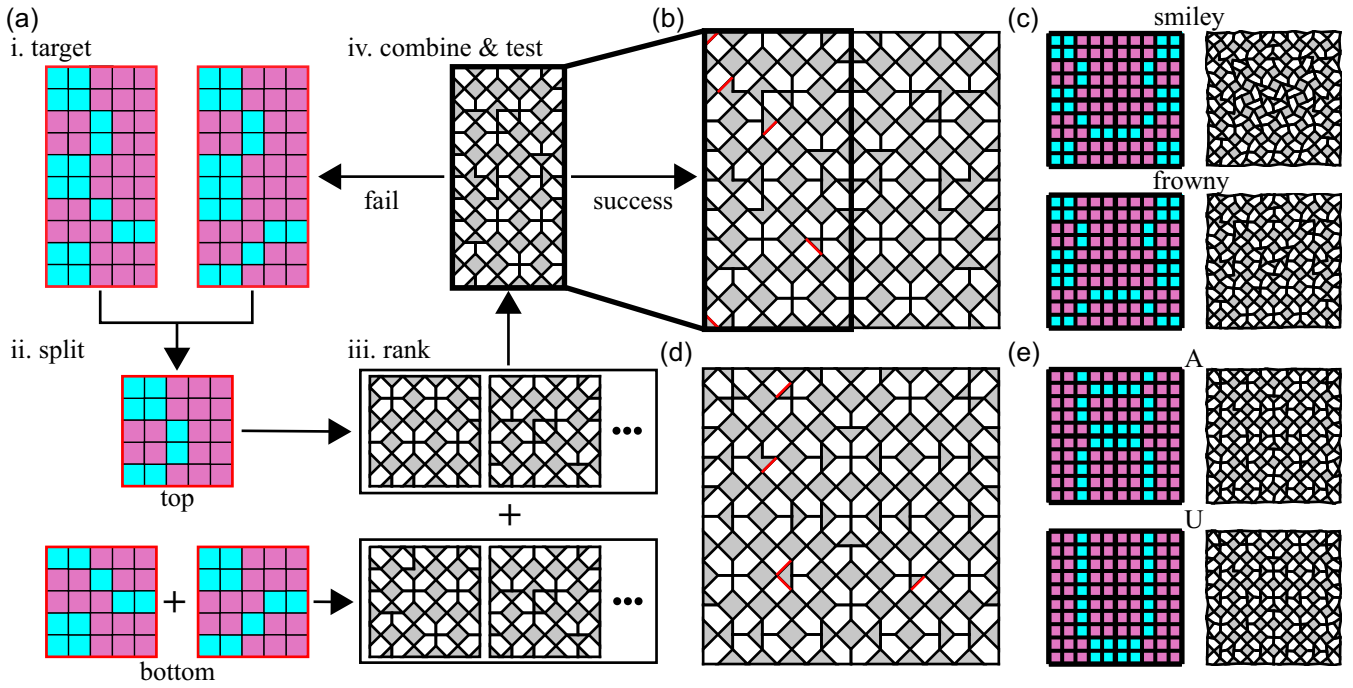

FIG. 5. Combining designs for larger target deformations. (a) Strategy for combining designs in four substeps. In step i, the two target modes (highlighted with a red border) are split (step ii) into separate sets of target modes for the top and bottom 5×5 designs. In step iii, the maximal cliques are ranked by the average capacity [Eq. (3)] for each set of target modes. In step iv, we progress iteratively through combinations of the highest-ranking designs until a combination with a sufficiently high capacity with respect to the two target modes of step (i) is found. This final design is mirrored and refined to form the final 10×10 design of (b). (b) The refined (defects are highlighted in red) 10×10 design for the target smiley and frowny zero modes. (c) The design of (b) supports two zero modes with structures resembling a smiley and frowny face. (d) Refined 10×10 design for the target A and U zero modes. (e) The design of (d) supports two deformation modes with structures resembling the letters A and U.

create two 10×10 metamaterials that each support two zero modes with spatially complex structures: one resembling a smiley and frowny face, and another resembling the letters A and U.

We demonstrate this strategy first for the smiley/frowny design target. We start by using symmetries to reduce the two targeted 10×10 mode structures into two 5×10 target modes [Figs. 5(a)(i)]; these can be split into a 5×5 top with one zero mode, and a 5×5 bottom with two zero modes [Fig. 5(a)(ii)]. As before, for each of these, we determine and rank all maximal cliques [Fig. 5(a)(iii)]. The nontrivial step is to combine the top and bottom: in general, combining two unit cells prohibits certain modes due to incompatible kinematic constraints at the interface. Therefore we iterate over the next-highest ranked combinations if the combined design is not satisfactory [Fig. 5(a)(iv)]. Using this approach, we find a successful 5×10 design; and using mirror symmetry, this yields a 10×10 design that features a mode structure resembling a smiley and frowny face [Figs. 5(b) and 5(c)]. For more details on our scaling-up method, see Appendixes D and F for pseudocode.

In addition to the smiley and frowny modes, we find seven superfluous modes [87]. We refine this design by strategically adding five extra rigid bars which reduces the number of superfluous modes to two [Fig. 4(g)]. We suggest that this design is an excellent starting point for further refinement to eliminate all spurious modes, e.g., by replacing a few building blocks with newly designed building blocks that only feature a single, complex deformation.

We follow the same procedure to find a design that features modes whose structures resemble the letters A and U, spelling Au for gold. Our best design features 11 superfluous modes, in addition to the two desired modes; by adding five rigid bars the number of superfluous modes is reduced to six [Fig. 5(e)]. Thus our design strategy allows us to find extremely rare designs with multiple desired modes within an otherwise intractable design space.

VI. DISCUSSION

Combining building blocks to create structures with desired properties is notoriously difficult as the design space is too vast to fully explore. Without access to underlying design rules, directly navigating such spaces with a strongly discontinuous objective function to find designs with desired properties is hopeless. In this paper, we introduced a general strategy to find ultrarare designs using pluripotency: a statistical measure that quantifies performance over a class of problems. In short, our approach exploits the hierarchical structure of pluripotency in the design space to generate a library of many high pluripotency designs. Subsequently, we select and refine from this library to reach the final, ultrarare design that satisfies the desired properties. Splitting the design problem into two parts, typically optimization

of (1) the geometry and (2) the continuous parametric design parameters, is a common strategy [64,66,89,90]. In contrast, our approach is gradient-free and leverages a multimodal property—pluripotency—which makes it uniquely well-suited for designing oligomodal combinatorial metamaterials.

Our approach opens up new exciting avenues for combinatorial design. For example, our approach can be readily applied to metamaterial design for a plethora of different building block designs, allowing for much faster exploration of the vast, largely unexplored design space of metamaterial geometries. Moreover, we foresee applications beyond the field of metamaterials. Self-assembling systems, for example, require the right set of building blocks to achieve the desired end geometry—which is hard without access to assembly rules [70,91,92]. Additionally, information processing in designer matter can be described by a set of hysteretic elements—how to order and tune interactions between these elements to achieve complex memory properties is an open challenge [93–95].

ACKNOWLEDGMENTS

We thank Daniel Acuña Ramirez for valuable discussions and feedback. This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative. C.C. acknowledges funding from the European Research Council under Grant No. ERC-2019-STG 852587 and from the Netherlands Organisation for Scientific Research under Grant Agreement No. VIDI 213131. M.D. acknowledges financial support from the European Research Council under Grant No. ERC-2019-ADV-H2020 884902.

DATA AVAILABILITY

The data that support the findings of this article are openly available [96–98].

APPENDIX A: METAMATERIAL

In this work, we focus on a combinatorial metamaterial built by tiling building blocks to form $k \times k$ unit cells which are periodically repeated to tile a larger $n \times n$ metamaterial [Fig. 6(a)]. This metamaterial is composed of a collection of rigid bars and hinges. Note that the building block is essentially a five-bar linkage. The building block thus features two zero modes: infinitesimal deformations that do not stretch any of the bonds up to first order. We choose the basis zero modes to be the CRS mode m_{CRS} and D mode $m_D(c)$ [Fig. 1(a)], where c is the orientation of the building block. Note that only the D mode depends on the orientation of the building block. This particular basis allows us (i) to easily distinguish between the trivial CRS deformation and other deformations in the metamaterial and (ii) to identify where to strategically place defects.

1. Statical determinacy

Generally, the structural integrity of a frame of N joints and N_B rigid bars is captured in the Maxwell-Calladine index

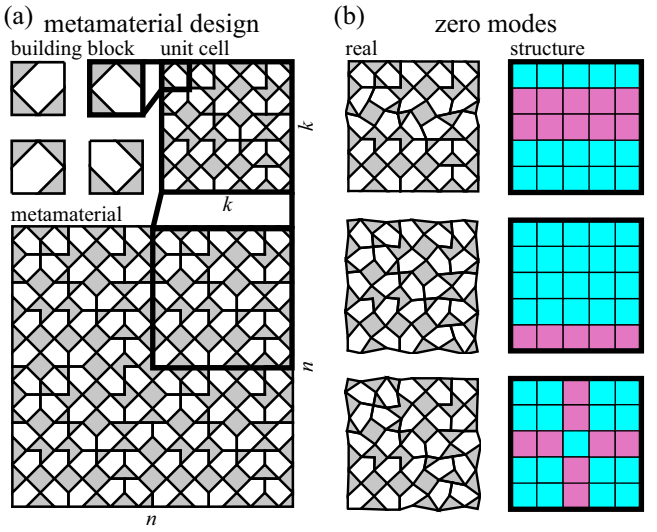

FIG. 6. Metamaterial design and zero modes. (a) Building blocks (top left) combine into a $k = 5$ unit cell (top right) to form a $n = 2$ metamaterial (bottom). (b) Examples of the three types of zero mode structures: (i) strip modes (top); (ii) edge modes (middle); (iii) global modes (bottom).

theorem in d dimensions [99,100]:

$$N_M - N_S = dN - N_B - 3 \equiv \mathcal{P}, \tag{A1}$$

where N_M and N_S are the number of nontrivial zero modes and states of self-stress respectively. We consider two-dimensional frames, i.e., $d = 2$. Because N_S is non-negative, the Maxwell count $\mathcal{P}$ is a lower bound for the number of modes, i.e., $N_M \geqslant \mathcal{P}$. Counter-intuitively, frames with intensive modes must be hyperstatic ($\mathcal{P} < 0$) for a sufficiently large periodic tiling of the unit cell frame. To see this, we note that for an $n \times n$ tiling of the unit cell frame, $\mathcal{P}(n)$ is a polynomial in n of maximum degree 2. To yield an intensive number of modes N_M we require the number of states of self-stress $N_S(n)$ to cancel all but the constant term of $\mathcal{P}(n)$. We note that states of self-stress are always localized with open boundary conditions, such that $N_S(n) \propto n^2$ in two dimensions. Thus we require $\mathcal{P}(n) \propto -n^2$ to yield oligomodal frames.

To see if our metamaterial satisfies this condition, we determine the metamaterial's Maxwell count:

$$\mathcal{P} = -(nk)^2 + 4nk + 2 \tag{A2}$$

for a $n \times n$ tiling of a $k \times k$ unit cell. These tilings are strictly hyperstatic for $nk > 2 + \sqrt{6} \approx 4.4$. In our paper, where we focus on unit cells of size $k = 5$, every tiling is hyperstatic. Thus, to yield intensive modes, the number of states of self-stress should scale as $N_S \propto (nk)^2 - 4nk$ and cancel the quadratic and linear terms of $\mathcal{P}$. We find that $N_M = an + b$, thus the quadratic term always cancels. The linear scaling and offset of the number of states of self-stress N_S with n then determines the exact values of a and b in $N_M(n)$ and depends on the orientations of the building blocks.

2. Calculating the number of zero modes

We determine the number of zero modes N_M of a design in two steps. First, we construct the compatibility matrix $\mathcal{C}$ of

the design's corresponding bar-joint framework. This matrix maps the deformations of the joints to elongations of the bars [100,101]. Zero modes are infinitesimal deformations of the joints that do not stretch any of the bonds up to first order. Mathematically, the dimension of the kernel or null-space of the compatibility matrix $\mathcal{C}$ corresponds to the number of zero modes. Thus, in our second step, we calculate this dimensionality using rank-revealing QR (rrQR) decomposition [82,83]. The time complexity of this algorithm is $\mathcal{O}(n^3)$ for $n \times n$ matrices.

To calculate the coefficients a and b from our mode-scaling relation [Eq. (1)], we use rrQR to calculate the number of modes $N_M(n)$ for $n \times n$ tilings of unit cells with open boundary conditions in the range $n \in \{1, 2, 3, 4\}$. We use $N_M(3)$ and $N_M(4)$ to determine the slope a and offset b. We use 1M 5×5 unit cells drawn from an uniform discrete distribution to obtain the distribution of b shown in Fig. 2(a).

To obtain Figs. 2(b) and 2(c), we selected at most 1000 designs per initial number of intensive modes b_I from the set of Monte Carlo sampled designs. For $b_\mathrm{I} = 5$ and $b_\mathrm{I} = 6$, there are fewer than 1000 designs available, instead we use only 311 and 8 designs, respectively. For each of the selected designs, we infer the number of strip modes a and intensive modes b for the 75 designs that differ from the selected design by a single building block mutation. We determine the ratio of mutated designs with b_M intensive modes to the total number of neighboring designs for each design and average over all designs for a given initial number of intensive modes b_I to obtain the probability density function $p(b_\mathrm{M}|b_\mathrm{I})$. Similarly, we define the ratio r for a design with b_I modes as the fraction of all mutated designs with $b_\mathrm{M} \leqslant b_\mathrm{I}$.

3. Determining the structure of a zero mode

A building block can deform with any linear combination of the two independent basis modes: $m = \alpha m_\mathrm{CRS} + \beta m_D$. We classify building blocks by these zero modes: if a block deforms with $\beta = 0$, it is a CRS block, and if a block deforms with $\beta \neq 0$, it is a D block. In earlier work, we showed that there are limitations on the spatial structure of zero modes in tilings of these building blocks: regions of adjacent CRS blocks must always be rectangular of shape [49].

To determine the structure of zero modes supported by a given design in terms of CRS and D blocks, a simple calculation of the kernel of the compatibility matrix $\mathcal{C}$ is no longer sufficient. Instead, we directly solve for the kinematic degrees of freedom of the building block: α and β. These kinematic degrees of freedom effectively describe the infinitesimal change in each of the five free interior angles of the building block to first order. The kinematic constraints between neighboring building blocks can be described as constraints of adjacent angles of building blocks. For more technical details on this representation of modes, we refer to our paper [49].

In short, we solve for each building block's kinematic constraints by composing a large integer matrix of all kinematic constraints between building blocks and use the Python package sympy [102] to find the null space of this matrix. This yields a set of rational vectors that form a basis for all valid zero modes in the design and translate directly to the kinematic degrees of freedom α and β. Thus this method allows us to determine the structure of zero modes for a given metamaterial design in terms of CRS and D blocks. The code for this is freely available on our public GitLab [96].

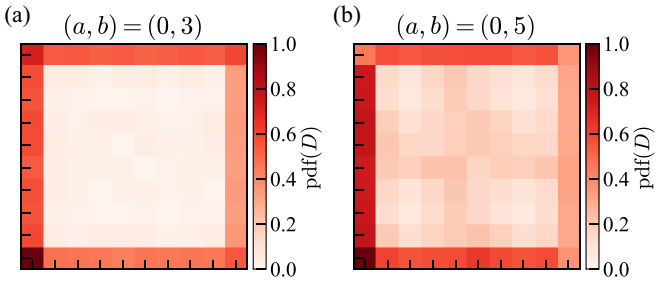

FIG. 7. Spatial distribution of the probility density function (pdf) of D blocks for 2×2 tilings of random 5×5 unit cells with $a = 0$ additive modes and $b = 3$ (a) or $b = 5$ (b) intensive modes.

4. Limitations to the structure of zero modes

In previous work, we have showed that there are limitations on the structure of zero modes [49]. Most importantly, we have shown that areas of adjacent CRS blocks must be rectangular of shape, i.e., their boundaries feature only convex corners. This constraint strongly limits the possible mode structures in our metamaterial. In particular, we distinguish between three types of zero modes: (i) strip modes, (ii) edge modes, and (iii) global modes [Fig. 6(b)]. Each of these mode types are defined by the spatial ordering of CRS and D blocks. Specifically, strip modes feature a horizontal or vertical strip of D blocks that spans the entire material sandwiched between two areas of CRS. Edge modes feature a strip of D blocks that borders the edge(s) of the material, the bulk are CRS blocks. Global modes feature D blocks throughout the entire material.

Additionally, these types of zero modes correspond to a change in the scaling of the number of nontrivial zero modes $N_M(n) = an + b$. Note that we exclude the three trivial zero modes corresponding to translation and rotation of rigid bodies in two dimensions. Strip-modes are translationally invariant in one direction, resulting in a linear increase of N_M and a contribution to the slope a. In contrast, edge and global modes are not translationally invariant: such modes correspond to an offset of N_M and thus contribute to the number of intensive modes b.

5. Distribution of edge and global modes

The codimension of the subspace of designs with b intensive modes appears multimodal in distribution, especially for larger b. This is most likely due to the different types of modes that contribute to b: edge modes and global modes. For low b, most intensive modes are edge modes [Fig. 7(a)]. As b increases, global modes become more prevalent [Fig. 7(b)]. The reason for this is twofold. First, edge modes consisting of a single line of D blocks require less building blocks with specific orientations than global modes that contain more D blocks (recall that the CRS mode is independent of the building block orientation). Thus random unit cells are more likely to contain edge modes than global modes. Second, the number of edge modes is limited by the number of edges of

the metamaterial. Thus, for sufficiently large number of intensive modes b, there is a higher probability of global modes. The codimension is related to the probability to prohibit an intensive mode by changing the orientation of a randomly selected building block. The type of mode influences this probability. In general, modes with more D blocks are easier to prohibit as the D mode is sensitive to the orientation of the building block. Additionally, building blocks at the edge are less kinematically restricted than building blocks in the bulk [49], making them more robust to changes of orientation.

6. Capacity of a set of modes

The capacity $C(\{M^j\}, \hat{M})$ [Eq. (3)] of a set of basis zero modes $\{M^j\}$ with respect to a target mode $\hat{M}$ is $S(M^*, \hat{M})$ [Eq. (2)], where $M^* = \sum w_j M^j$ is a linear combination of the modes, and the set of weights $\mathbf{w} = \{w_j\}$ maximizes S. To find this set of weights, we use constraint programming.

Formally, we define our constraint satisfaction problem as follows. We have a set of integer variables $\mathbf{w} = \{w_j\}$ and define a set of inequality constraints $\mathbf{l} = \{l_i\}$ for each site i where $\exists_j \beta_i^j \neq 0$ holds as

$$l_i = \begin{cases} \sum_j w_j \beta_i^j \neq \delta_i, & \text{if } \hat{m}_i = \text{D} \\ \left\{ \sum_j w_j \beta_i^j \geqslant -\delta_i, \sum_j w_j \beta_i^j \leqslant \delta_i \right\}, & \text{if } \hat{m}_i = \text{CRS} \end{cases}, \tag{A3}$$

where β_i^j is the coefficient of deformation $m_D(c_i^j)$ of mode M^j at site i and δ_i is a non-negative integer variable. If $\beta_i^j = 0$ is true for every j, site i in M^* is always a CRS block regardless of $\mathbf{w}$ and is thus excluded as a constraint. We use a constraint programming solver that uses satisfiability methods [103]. The solver aims to find a set of weights $\mathbf{w}$ and δ that satisfy the constraints while minimizing the objective function

$$O = \sum_i \delta_i. \tag{A4}$$

Note that because δ_i is a non-negative integer, this objective function is minimal if and only if $\delta_i = 0$ for all i. The upper constraint in Eq. (A3) ensures site i is a D block if $\delta_i = 0$. The lower constraint ensures site i is a CRS block if $\delta_i = 0$. The underlying idea is that by minimizing O, the algorithm tries to find a set $\mathbf{w}$ for which most variables satisfy $\delta_i = 0$. When $\delta_i = 0$ for all i, the mode M^* has the same kind of block, CRS or D, at site i as the target mode $\hat{M}$. The final similarity is then $S(M^*, \hat{M})$ [Eq. (2)] with $M^* = \sum_j w_j M^j$. The code we used to calculate the capacity is publicly available [97]. Pseudocode of this algorithm is shown in algorithm 1.

7. Randomized target deformations

To determine the pluripotency P of a design or a set of zero modes, we calculate the average capacity C [Eq. (3)] for a set of randomized target modes $\hat{M}$. We define our target modes $\hat{M}$ in terms of their structure: the spatial distribution of CRS and D blocks. However, our metamaterial has limitations regarding the mode structures it can achieve. In previous work, we demonstrated that any valid mode must be comprised of rectangular patches of adjacent CRS blocks [49]. Thus, to assess our designs fairly, we cannot simply generate any random distribution of CRS and D blocks.

Instead, we generate viable target modes by generating a horizontal and vertical strip of D blocks within a background of CRS blocks. The position and width of these strips are randomly drawn from a uniform distribution. Where the two strips overlap, we replace the D blocks with CRS blocks. This approach ensures that there are only rectangular patches of adjacent CRS blocks present. Using this approach, we generate 200 unique target modes. The 200 possible target deformations are shown in Fig. 8.

APPENDIX B: PREDICTING THE NUMBER OF INTENSIVE MODES

In contrast to our previous work [61], we now ask a neural network to predict the number of intensive modes b. We train our networks on 5×5 designs obtained from Monte Carlo sampling of the space. We note that determining the mode-scaling of these designs is computationally expensive, with a time complexity that grows cubically with input size [82]. Before training, we preprocess the data.

1. Preprocessing

The configurational data of the unit cell designs is preprocessed to a pixelated representation [see Fig. 9(a)] for input to the neural network. We found that representing the unit cell designs in this pixelated representation improved convergence during training and better performance of the trained networks over the validation sets. We believe that this is due to the orientations of the building blocks being clearly represented visually as opposed to simply representing the orientations as integers in a matrix. Because each building block is represented as a 2×2 square, we can choose where the convolutional layer applies its filters more finely. This allows us to let the networks "see" only the interactions between building blocks. Additionally, we use periodic padding to pad the designs with an extra one pixel wide layer to allow the network to see interactions between building blocks with periodic boundary conditions.

2. Neural network architecture

Because our designs are spatially structured and local interactions between building blocks drive compatible deformations, CNNs are well-suited for predicting the number of intensive modes b. The CNNs used in this work are composed of three convolution layers for feature extraction, which are connected to a fully-connected hidden layer, which in turn is connected to a single-neuron output layer. Specifically, the first convolution layer consists of twenty 2×2 filters with a stride of $(2, 2)$. As the convolution starts in the upper left corner of the input image, the network convolves only 2×2 *plaquettes* between four building blocks. Each plaquette contains a black pixel if one of the four constituent building blocks is oriented such that it has its diagonal interior angle within that plaquette [Fig. 9(a)]. As such, each plaquette contains information on which building blocks share adjacent diagonal corners and allow for possible compatible deformations of those corners. We conjecture that restricting the

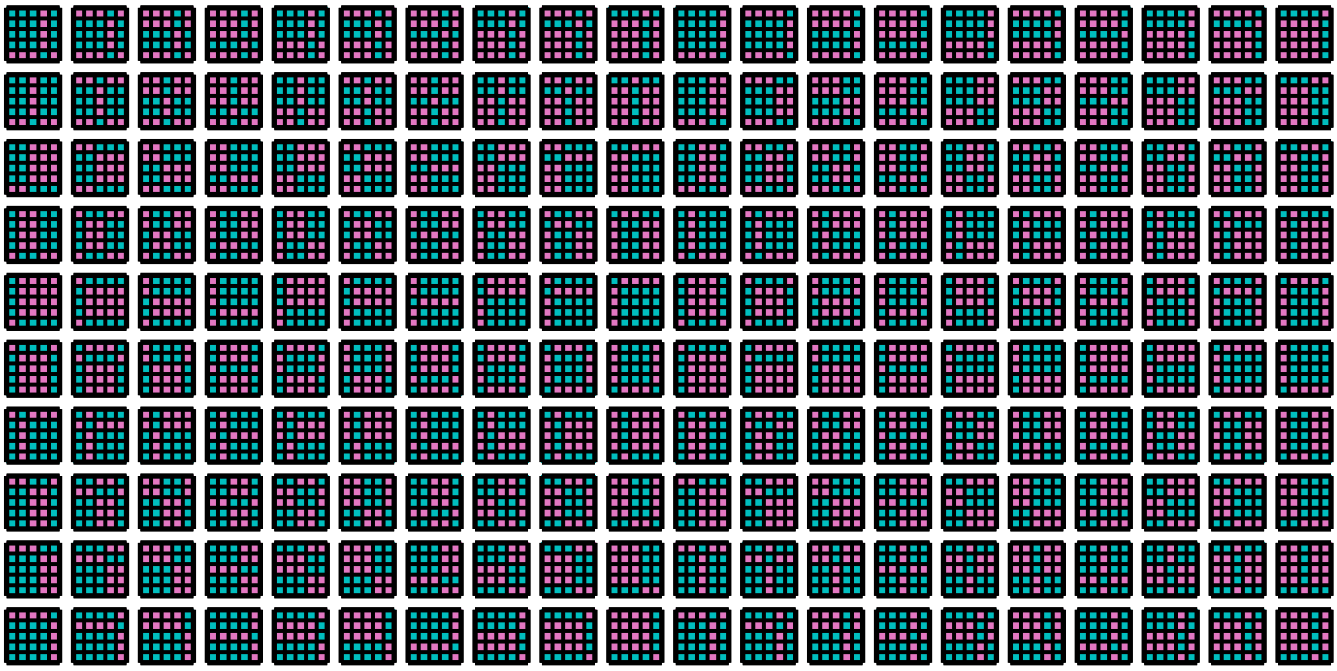

FIG. 8. All 200 possible global target deformations generated using the method as described in the Appendix A 7.

network to see only these plaquettes helps achieve a better and more robust performance.

The second and third convolution layers have 80 and 160 2×2 filters, respectively, with a stride of (1, 1). After each convolution operation, we add a trainable bias vector and apply a ReLu activation function on each element of the convolved images. Note that we do not use pooling operations in-between convolution layers. After the third convolution layer, we flatten the convolved images and fully connect this vector to a hidden layer of 1000 neurons. Again we add a bias vector and apply the ReLu activation function. The final layer consists of only a single neuron, and we do not apply an activation function. We take the single output neuron to be b_{CNN}, which we aim to be as close to the true b of any input design as possible. We use Jax [104] to code our networks.

3. Training the convolutional neural networks

To train the CNNs to predict the number of intensive modes b_{CNN}, we generate a set of unit cell designs $\mathbf{X} = \{X_i\}$ and their respective number of intensive modes $\mathbf{b} = \{b_i\}$. The

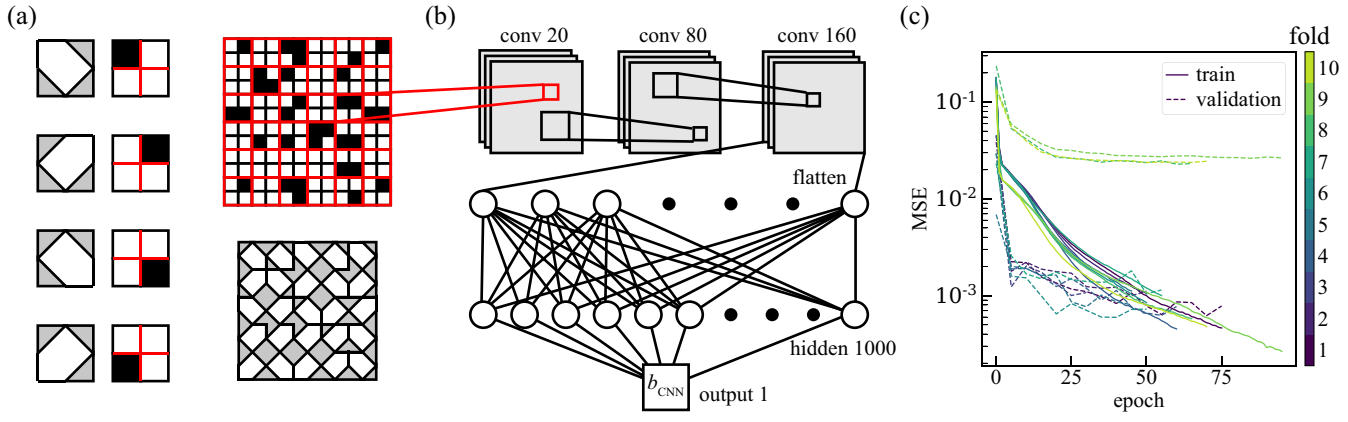

FIG. 9. Convolutional neural network (CNN) for metamaterial prediction. (a) To feed our metamaterial designs into a neural network, we represent our designs as a black-and-white image. Building blocks are represented as 2×2 plaquettes of pixels, one black and three white (left). 5×5 designs (bottom right) translate to 10×10 pixel images. Additionally, we pad these images using periodic boundary conditions with one additional layer of pixels, so that we end up with a 12×12 pixel image (top right). (b) The pixel image of (a) forms the input for our CNN, which consists of three convolutional layers, a single hidden layer and a single output node. (c) The training (solid line) and validation (dashed line) mean squared error (MSE) over the training epochs for each fold (colorbar) in our tenfold cross-validation. Note that we used an early stopping condition to prevent overfitting, resulting in different number of training epochs for the folds.

combination of these designs and their corresponding numbers of intensive modes is referred to as the training set $\mathbf{D}_t = (\mathbf{X}, \mathbf{b})$. The generated data is then divided into a training (85%) and a test (15%) set. Specifically, the training set contains 848898 samples and the set set contains 149982 samples. Our CNNs are trained on imbalanced datasets—designs with a large b become increasingly hard to find using Monte Carlo sampling [Fig. 2(a)]. Both the training and test set follow the same distribution of the number of intensive modes b; the majority of samples have $b = 1$, while there are only four samples with $b = 6$ in the training set.

We train the CNN to minimize the mean squared error (MSE) between the CNN's prediction b_{CNN} and the true b using the Adam optimization algorithm [105]. We use tenfold cross-validation to validate the robustness of our network architecture and training procedure. The network with the lowest mean squared error (MSE) over the validation set is selected to be the primary network to use. Specifically, we use a learning rate of 0.0005, train the network for 100 epochs and use a batch size of 256. Additionally, we use L2 regularization on the weights and biases. The architecture of our CNN is shown schematically in Fig. 9(b). The training process for each fold is shown in Fig. 9(c).

APPENDIX C: PROSPECTING PLURIPOTENT DESIGNS

We employ a genetic algorithm (GA) to explore our CNN beyond its training scope and to prospect highly pluripotent designs. Specifically, we utilize a GA where the fitness function [Eq. (5)] is estimated by a trained CNN. The goal of the GA is to achieve a target number of intensive zero modes b_T. To achieve this goal, the GA iteratively generates a population of designs (the generation) in three steps: (i) sampling, (ii) fitness evaluation, and (iii) update. Below, we give an overview of these three steps.

In the first step, a fixed number of candidate designs is randomly drawn from the discrete design space. In GA terminology, this set of designs (population) is referred to as generation 0. Only the very first generation is generated in this manner. Second, we score and rank the designs based on their fitness f [Eq. (5)]. This fitness is maximal when the CNN's prediction is equivalent to the target number of intensive modes b_T. Finally, we use the ranking of designs based on the fitness to generate a new population of designs—the next generation. To efficiently explore the design space, the GA combines and mutates designs. To this end, we employ several standard GA techniques. We select a group of designs from the initial population based on a three-way random tournament selection, and we always include the design with the highest fitness (elitism). This group of designs forms the "parents." From this set of parents, we combine designs using a custom crossbreeding scheme (see below), and we clone to create an additional set of designs. This set of designs also undergoes random mutations of building blocks and produces the "children." The combination of parents and children then forms the next generation, maintaining the same size as the previous generation. This procedure of fitness evaluation and generation of a new population of designs is repeated until the fitness is maximized or a predefined criterion is met.

In short, the GA is able to combine designs to generate new designs with an average b, after which random mutations eventually allow the GA to reach a high b. As GAs are known to go through a lot of generations, the computational bottleneck of such algorithms is typically the calculation of the fitness for each new generation. To save computation time, we use a CNN to calculate the fitness f [Eq. (5)].

In more detail, our genetic algorithm has a population size of 100, of which 29 are selected to be parents using a three-way tournament selection. Additionally, the fittest candidate in the population is automatically selected to be one of the parents, so that we have a total of 30 parents. To create a new generation of 100 designs, we require 70 children. Of those 70 children, half are created using cross-breeding of randomly selected pairs of parents. To combine the design of two parents to create a new child, we use a Gaussian filter to filter a random binary 5×5 mask, which we then binarize again to create a mask where half of the elements are 0 and half are 1. This mask is then used to take part of the two parents and combine them to create a new child [Fig. 10(a)]. The reason we use this method over more standard methods, such as k-point crossover or uniform crossover, is that we believe local clusters of building blocks are important for intensive zero modes. The other 35 children are taken by cloning randomly selected parents. All building blocks in the children designs have a chance of 10% to mutate to a different orientation. Each of these different orientations are equally likely to be selected. The combination of the parents and children after mutation form the new generation of designs.

1. GA exploits nonlocal structure

To understand how the GA explores design space, we investigate the two key exploration techniques of our GA. Underlying the evolution of generations of designs are two main processes: (i) cloning and (ii) crossbreeding [Fig. 10(b)]. Our GA thus explores the design space in two ways: (i) the cloned designs undergo local mutations. This is local exploration of the design space surrounding the cloned parent design. (ii) The crossbred designs are combinations of two parent designs on top of local mutations. This is nonlocal exploration of the design space.

The combination of both local and nonlocal exploration are crucial for the success of the GA. To illustrate this, we determine the minimal distance between between the design X with the highest fitness of generation $g + 1$ and every other design Y in generation g as

$$d_{\min} = \min_{\{Y\}} \left(\sum_{i,j} 1 - \delta(X_{i,j}, Y_{i,j}) \right), \tag{C1}$$

where $\delta(x, y)$ is the Kronecker delta function and $X_{i,j}$ is the orientation of the building block at site (i, j) in design X. Eq. (C1) thus captures the number of different building blocks between design X of generation $g + 1$ and the design Y in generation g that is most similar to X. Nonlocal exploration primarily plays a role for early generations when $b_{\mathrm{CNN}}(g)$ is likely to be small, which allows the GA to find designs of a higher b_{CNN} (Fig. 11(c)). For larger b_{CNN}, in later generations, local exploration is key.

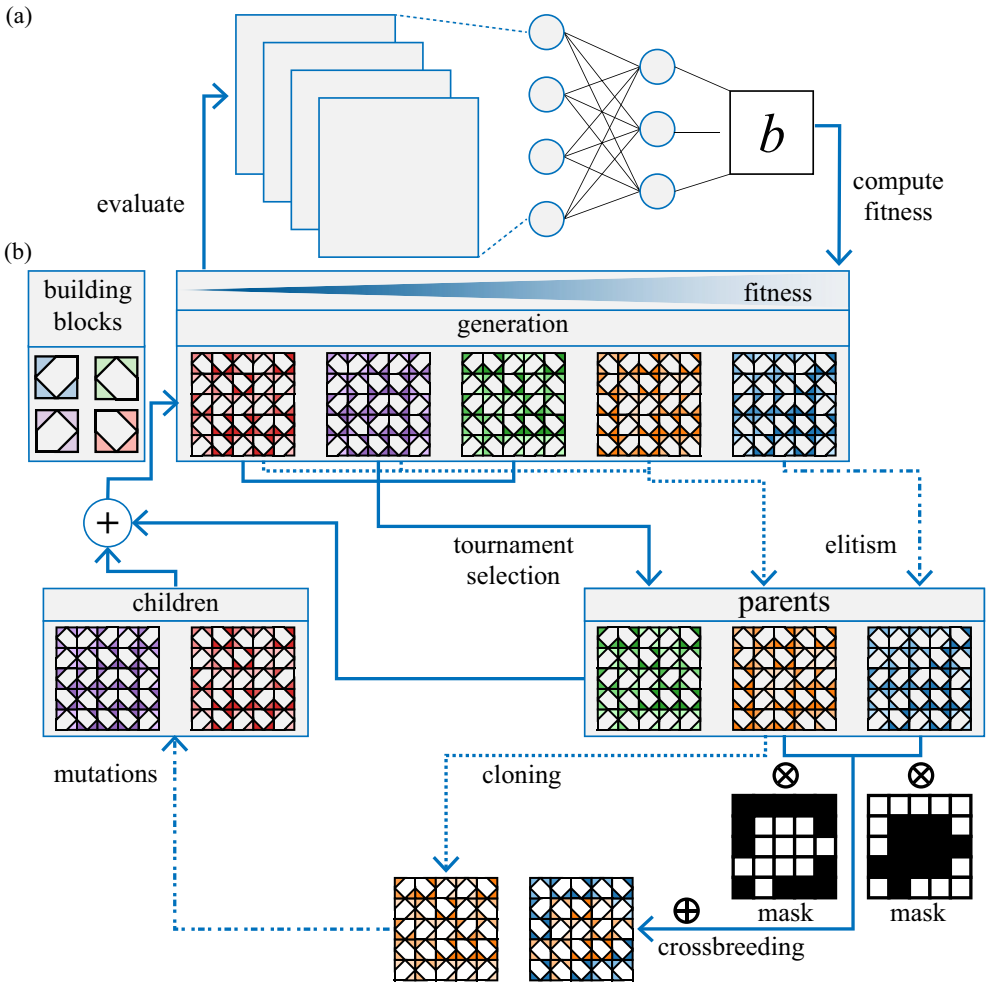

FIG. 10. Schematic representation of our genetic algorithm. (a) A CNN calculates the number of intensive modes b_{CNN} for each design in the generation. (a) A generation of designs (top) is ranked based on the fitness f. Using a three-way tournament selection the algorithm selects designs from the generation that, together with the fittest design in the generation, form the parents (right). From the parents, we generate new designs in two ways: cloning and crossbreeding (bottom). In cloning, we simply make a copy of the parent design chosen at random. In crossbreeding, we randomly select two parents that we combine using two-dimensional masks (black-and-white matrices). Additionally, building blocks in these newly generated designs have the chance to mutate into new orientations. The mutated designs form the children (left) who, together with the parents, form the next generation. This iterative procedure continues until a sufficiently high fitness or stopping condition is reached.

Intuitively, the GA is able to efficiently explore the design space by first crossbreeding designs to quickly find designs with a reasonably high number of intensive modes b_{CNN}. This is most likely a consequence of the fact that most designs with a low number of intensive modes feature deformations that are localized on the edge of the material. Such edge modes are less sensitive to building block mutations than global modes and thus crossbreeding is more likely to combine designs that feature edge modes. Moreover, as the number of intensive modes b increases the probability to inhibit any mode by changing orientations increases, such that the crossbreeding is more likely to result in a net conservation or decrease of b. After $b = 5$, the GA appears to rely on cloning to locally explore around an ensemble of designs with high b to find rare $b + 1$ designs. Thus, the GA is able to efficiently generate designs that feature a large number of intensive modes b thanks to both nonlocal and local exploration, resulting in successful runs that generally converge.

2. Random walks

To both obtain starting designs for the random walks and compare the evolution of GA designs, we perform and keep track of a hundred GA runs with $b_T = 7$. We start a hundred random walks from the final hundred designs with the highest fitness, thus the starting $b(s = 0)$ varies from $5 \leqslant b \leqslant 7$.

3. Comparison to other methods

Our GA is able to find ultra-rare designs with a large number of intensive modes b, that is not possible using random search as designs with high b become exponentially more rare

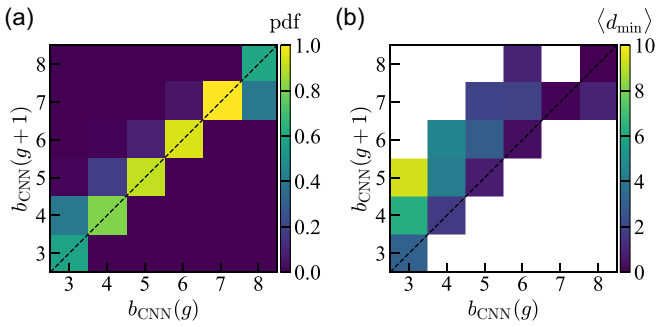

FIG. 11. GA design space exploration. (a) Pdf for the design with the highest fitness in a GA run to transition from $b_{\mathrm{CNN}}(g)$ to $b_{\mathrm{CNN}}(g+1)$ for iterative generations. For low $b_{\mathrm{CNN}}(g)$ the GA is more likely to transition to a higher b_{CNN}. For high $b_{\mathrm{CNN}}(g)$ the GA struggles to quickly find designs with a larger $b_{\mathrm{CNN}}(g+1)$. Note that $b_{\mathrm{CNN}}(g)$ can never transition to a lower $b_{\mathrm{CNN}}(g+1)$ as the GA always takes the design with the highest b_{CNN} to the next generation. (b) The average minimal distance $\langle d_{\min} \rangle$, where $\langle . \rangle$ denotes the average over an ensemble of GA runs, for the fittest design in generation $g+1$ with $b_{\mathrm{CNN}}(g+1)$ as a function of $b_{\mathrm{CNN}}(g)$ of the fittest design in generation g. As $b_{\mathrm{CNN}}(g)$ increases, the distance $\langle d_{\min} \rangle$ between the fittest design in the next generation $g+1$ and all designs in generation g decreases.

with b. Alternatively, one could try to exploit the hierarchical design space structure through a hill-climbing method. While this approach succeeds sometimes, it fails an exponentially larger fraction of the time for increasing target number of intensive modes b_T [Figs. 12(a) and 12(b)]. This results in a

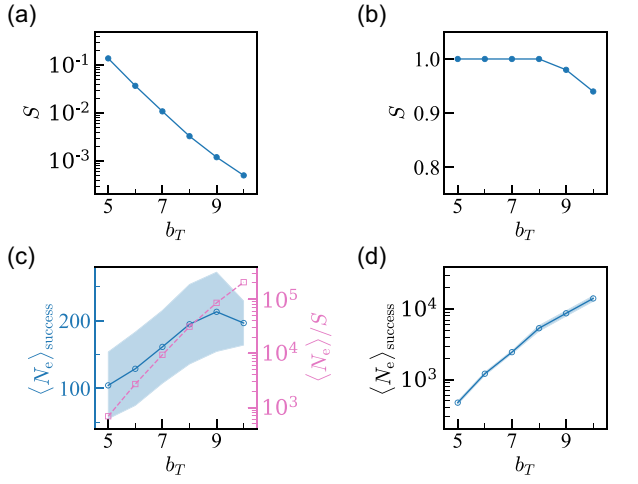

FIG. 12. Comparison of hill-climbing and genetic algorithm. (a) Success rate S of 10 000 hill-climbing runs to reach target number of intensive modes b_T decreases exponentially. (b) The success rate S of 10000 GA runs to reach b_T. (c) The average (blue circles) and standard deviation (blue shaded area) of the number of evaluations of the fitness function f for successful runs $\langle N_e \rangle_{\mathrm{success}}$ to reach b_T using hill climbing. The average number of evaluations per successive run $\langle N_e \rangle$ S (pink squares) increases exponentially with b_T. (d) The average (blue circles) and standard deviation (shaded area) of the number of evaluations of the fitness function f for successful runs $\langle N_e \rangle_{\mathrm{success}}$ increases exponentially with b_T.

larger average number of evaluations of the fitness function f for large b_T to make a successful run [Figs. 12(c) and 12(d)]. Compared to state-of-the-art generative methods, such as variational autoencoders (VAE), generative adversarial neural networks (GANs), and normalizing flows, our method allows for extrapolation outside the scope of the training set. These generative methods all aim to approximate the (unknown) underlying probability distributions of the training set—without examples such methods are unable to generate designs with the desired property.

APPENDIX D: COMBINING AND SELECTING DESIGNS FOR TARGET DEFORMATIONS

Here, we describe our method to combine and select from a set of 5×5 designs to form larger metamaterials that deform close to desired target deformations. We start from a set of designs generated in step (i) and compute the mode structures. We aim to throw away edge modes. We do this by checking the location of CRS sites; if the entire 4×4 inner square of the 5×5 mode consists of CRS blocks, we assume it is an edge mode and disregard it.

Next, we determine which modes are *compatible* with each other. Here, compatible means that there exists a design that can feature both modes. Modes are compatible if (1) there are no overlapping D sites or (2) if for all overlapping D sites, the building block orientations from the original designs are the same. This means that all modes that originate from the same design are compatible, as they should be. Surprisingly, modes that originate from different designs can also be compatible. By taking the building block orientations of the D sites for both modes, we can create a design that features both modes, as the orientation of CRS blocks is irrelevant. This procedure yields a set of modes and pairs of modes that we label compatible. We represent this as a graph with nodes (modes) and undirected edges (compatible).

To explore our set of modes for combinations that deform close to a target deformation, we search the space of maximal cliques. Such cliques hold the largest set of compatible modes, thus providing the design with the most deformational freedom. We calculate the set of maximal cliques using the Bron-Kerbosch algorithm with vertex ordering by first calculating the degeneracy ordering of the graph [88,106]. For any given set of designs, we need to perform this calculation only once and save the graph and maximal clique for subsequent searches for target deformations.

To determine if a maximal clique of modes deforms closely to target deformations, we calculate the capacity C [Eq. (3). This score is calculated using a constraint programming solver that uses satisfiablity methods [103] as described in Appendix A 6.

To design larger metamaterials, such as the 10×10 designs of Figs. 5(b) and 5(d), we split the larger target deformation into local 5×5 deformations. We rank all maximal cliques based on their cumulative capacity C for each 5×5 subset. Then, we take a greedy search approach and combine designs with the highest capacity for each 5×5 subset and calculate the total capacity for the combined design. If the total score in unsatisfactory (below a predefined threshold), we iteratively try the next best designs for each 5×5

ALGORITHM 1. Calculate capacity of set of modes with respect to target mode structure

Input: $\{M^j\}$ ▷ set of zero modes
$\hat{M}$ ▷ target mode structure
Output: C ▷ capacity
Initialize integer weight variables **w**
procedure GENERATECONSTRAINTS
  l = {} ▷ initialize empty list of constraints
  for $\hat{m}_i$ in $\hat{M}$ **do**
    if $\exists_j \beta_i^j \neq 0$ **then**
      if $\hat{m}_i$ = D **then**
        initialize non-negative integer variable δ_i
        $l_i \leftarrow \sum_j w_j \beta_i^j \neq \delta_i$
      end if
      if $\hat{m}_i$ = CRS **then**
        initialize non-negative integer variable δ_i
        $l_i \leftarrow \left\{ \sum_j w_j \beta_i^j \geqslant -\delta_i, \sum_j w_j \beta_i^j \leqslant \delta_i \right\}$
      end if Add l_i to **l**
    end if
  end for
end procedure
GENERATECONSTRAINTS
$O \leftarrow \sum \delta_i$ ▷ define objective function
w, $\delta \leftarrow$ *CPSATSolver* $(O, \mathbf{l})$ ▷ call constraint programming solver to minimize O while satisfying **l**
$M^* \leftarrow \sum_j w_j M^j$
set $\hat{N}_{\text{CRS}}$ equal to number of CRS blocks in $\hat{M}$
$C \leftarrow S(M^*, \hat{M}) - \hat{N}_{\text{CRS}}$ ▷ Calculate S using Eq. (2) in the Main Text

subset until we find a satisfactory total score. Pseudo-code of the algorithm is shown in Algorithm 2.

APPENDIX E: PROHIBITING UNDESIRED ZERO MODES

To prohibit undesired zero modes, we strategically introduce additional rigid bars to the metamaterial structure. To determine the placement of these bars, we analyze the modal structure of the desired and undesired zero modes. In particular, if the undesired mode contains a D block where the desired modes feature CRS blocks, we add a rigid bar across the diagonal to transform the pentodal building block shape to a square shape. This prohibits the building block from deforming with the D mode, effectively prohibiting the undesired mode without introducing any new (undesired) zero modes.

APPENDIX F: PSEUDOCODES

Pseudocode for calculating the capacity C [Eq. (3)]] is shown in Algorithm 1. Pseudocode for the extracting step of step (ii) of our design approach, including the step to scale-up to larger designs, is shown in Algorithm 2.

ALGORITHM 2. Find design that matches desired mode structures.

Input: X ▷ set of pluripotent generated 5 × 5 base designs
$\{\hat{M}\}$ ▷ set of target deformations, can be multiples of 5 × 5
Output: D ▷ design
procedure MODELIST ▷ Calculate zero modes for each design
  M $\leftarrow$ {} ▷ empty list of modes
  for X in **X do**
    $\{M\} \leftarrow$ *CalculateModes*(X) ▷ set of modes
    Add $\{M\}$ to **M** ▷ Add set to list
  end for
end procedure
procedure FILTEREDGEMODES ▷ check for edge modes
  for M in **M do** ▷ for each mode in list
    if inner 4 × 4 square of M is all CRS blocks **then**
      Remove M from **M**
    end if
  end for
end procedure

ALGORITHM 2. (*Continued.*)

procedure DETERMINECOMPATIBILITY
  $A \leftarrow \mathbf{0}$ ▷ empty adjacency matrix
  $G \leftarrow \{\mathbf{M}, A\}$ ▷ graph with modes as nodes
  for each M^i in **M do**
    for each M^j in **M do**
      if M^i and M^j originate from same X **then**
        Add edge (i, j) to A
      else if $Compatible(M^i, M^j)$ **then** ▷ check if compatible (see Appendix D)
        Add edge (i, j) to A
      end if
    end for
  end for
end procedure
procedure FINDMAXIMALCLIQUES
  $G_D \leftarrow DegeneracyOrdering(G)$ ▷ degeneracy ordering of G
  $\mathbf{U} \leftarrow BronKerbosch(G_D)$ ▷ list of maximal cliques using Bron-Kerbosch algorithm
end procedure
procedure RANKCLIQUES
  $\mathbf{R} \leftarrow \{\}$ ▷ empty list of capacities
  $\hat{\mathbf{T}} \leftarrow RandomizedTargets()$ ▷ generate randomized target deformation modes
  for each U in **U do**
    $N_U \leftarrow Size(U)$ ▷ number of modes in clique
    $r = 0$ ▷ cumulative capacity
    for each $\hat{T}$ in $\hat{\mathbf{T}}$ **do**
      $r \leftarrow r + C(U, \hat{T})$ ▷ calculate capacity C using Al. 1
    end for
    $r \leftarrow r/N_U$ ▷ average capacity
    add r to **R**
  end for
  Order **U** from high to low based on **R**
end procedure
procedure DESIGNMETAMATERIAL
  $D \leftarrow \mathbf{0}$ ▷ empty design
  ϵ ▷ threshold
  while cumulative capacity C is below ϵ **do**
    Iterate through ranked cliques **U** per 5×5 subset
    Turn each clique into 5×5 design
    Combine 5×5 designs into design D
    Calculate cumulative C of D with respect to each target mode in $\{\hat{M}\}$
  end while
end procedure
MODELIST
FILTEREDGEMODES
DETERMINECOMPATIBILITY
FINDMAXIMALCLIQUES
RANKCLIQUES
DESIGNMETAMATERIAL

[1] T. E. Bruns and D. A. Tortorelli, Topology optimization of non-linear elastic structures and compliant mechanisms, Comput. Methods Appl. Mech. Eng. **190**, 3443 (2001).

[2] O. Sigmund and K. Maute, Topology optimization approaches: A comparative review, Struct. Multidisc. Optim. **48**, 1031 (2013).

[3] M. Wang, X. Hu, D. N. Beratan, and W. Yang, Designing molecules by optimizing potentials, J. Am. Chem. Soc. **128**, 3228 (2006).

[4] C. P. Goodrich, E. M. King, S. S. Schoenholz, E. D. Cubuk, and M. P. Brenner, Designing self-assembling kinetics with differentiable statistical physics models, Proc. Natl. Acad. Sci. **118**, e2024083118 (2021).

[5] E. Medina, C. H. Rycroft, and K. Bertoldi, Nonlinear shape optimization of flexible mechanical metamaterials, Extreme Mech. Lett. **61**, 102015 (2023).
[6] G. Bordiga, E. Medina, S. Jafarzadeh, C. Bösch, R. P. Adams, V. Tournat, and K. Bertoldi, Automated discovery of reprogrammable nonlinear dynamic metamaterials, Nat. Mater. **23**, 1486 (2024).
[7] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, Automatic chemical design using a data-driven continuous representation of molecules, ACS Cent. Sci. **4**, 268 (2018).
[8] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. **378**, 686 (2019).
[9] D. C. Elton, Z. Boukouvalas, M. D. Fuge, and P. W. Chung, Deep learning for molecular design–a review of the state of the art, Mol. Syst. Des. Eng. **4**, 828 (2019).
[10] O. Khatib, S. Ren, J. Malof, and W. J. Padilla, Deep learning the electromagnetic properties of metamaterials–a comprehensive review, Adv. Funct. Mater. **31**, 2101748 (2021).
[11] K. Guo, Z. Yang, C.-H. Yu, and M. J. Buehler, Artificial intelligence and machine learning in design of mechanical materials, Mater. Horiz. **8**, 1153 (2021).
[12] X. Zheng, X. Zhang, T.-T. Chen, and I. Watanabe, Deep learning in mechanical metamaterials: From prediction and generation to inverse design, Adv. Mater. **35**, 2302530 (2023).
[13] L. Wang, Y.-C. Chan, F. Ahmed, Z. Liu, P. Zhu, and W. Chen, Deep generative modeling for mechanistic-based learning and design of metamaterial systems, Comput. Methods Appl. Mech. Eng. **372**, 113377 (2020).
[14] L. Zheng, S. Kumar, and D. M. Kochmann, Data-driven topology optimization of spinodoid metamaterials with seamlessly tunable anisotropy, Comput. Methods Appl. Mech. Eng. **383**, 113894 (2021).
[15] J.-H. Bastek, S. Kumar, B. Telgen, R. N. Glaesener, and D. M. Kochmann, Inverting the structure–property map of truss metamaterials by deep learning, Proc. Natl. Acad. Sci. USA **119**, e2111505119 (2022).
[16] B. Telgen, O. Sigmund, and D. M. Kochmann, Topology optimization of graded truss lattices based on on-the-fly homogenization, J. Appl. Mech. **89**, 061006 (2022).
[17] B. Deng, A. Zareei, X. Ding, J. C. Weaver, C. H. Rycroft, and K. Bertoldi, Inverse design of mechanical metamaterials with target nonlinear response via a neural accelerated evolution strategy, Adv. Mater. **34**, 2206238 (2022).
[18] N. K. Brown, A. P. Garland, G. M. Fadel, and G. Li, Deep reinforcement learning for the rapid on-demand design of mechanical metamaterials with targeted nonlinear deformation responses, Eng. Appl. Artif. Intell. **126**, 106998 (2023).
[19] C. S. Ha, D. Yao, Z. Xu, C. Liu, H. Liu, D. Elkins, M. Kile, V. Deshpande, Z. Kong, M. Bauchy *et al.*, Rapid inverse design of metamaterials based on prescribed mechanical behavior through machine learning, Nat. Commun. **14**, 5765 (2023).
[20] A. D. Keefe and J. W. Szostak, Functional proteins from a random-sequence library, Nature (London) **410**, 715 (2001).
[21] M. H. Hecht, A. Das, A. Go, L. H. Bradley, and Y. Wei, De novo proteins from designed combinatorial libraries, Protein Sci. **13**, 1711 (2004).
[22] P.-S. Huang, S. E. Boyken, and D. Baker, The coming of age of de novo protein design, Nature (London) **537**, 320 (2016).
[23] D. Rouvray, Isomer enumeration methods, Chem. Soc. Rev. **3**, 355 (1974).
[24] S. Otto, R. L. Furlan, and J. K. Sanders, Dynamic combinatorial chemistry, Drug Discovery Today **7**, 117 (2002).
[25] B. Pirard, The quest for novel chemical matter and the contribution of computer-aided de novo design, Expert Opinion on Drug Discovery **6**, 225 (2011).
[26] U. Feldkamp and C. M. Niemeyer, Rational design of dna nanoarchitectures, Angew. Chem., Int. Ed. **45**, 1856 (2006).
[27] Y. Dong, C. Yao, Y. Zhu, L. Yang, D. Luo, and D. Yang, Dna functional materials assembled from branched dna: Design, synthesis, and applications, Chem. Rev. **120**, 9420 (2020).
[28] S. A. Cummer, J. Christensen, and A. Alù, Controlling sound with acoustic metamaterials, Nat. Rev. Mater. **1**, 16001 (2016).
[29] C. Coulais, E. Teomy, K. De Reus, Y. Shokef, and M. Van Hecke, Combinatorial design of textured mechanical metamaterials, Nature (London) **535**, 529 (2016).
[30] B. Jenett, C. Cameron, F. Tourlomousis, A. P. Rubio, M. Ochalek, and N. Gershenfeld, Discretely assembled mechanical metamaterials, Sci. Adv. **6**, eabc9943 (2020).
[31] F. Zangeneh-Nejad, D. L. Sounas, A. Alù, and R. Fleury, Analogue computing with metamaterials, Nat. Rev. Mater. **6**, 207 (2021).
[32] C. Blum and A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM comput. surv. (CSUR) **35**, 268 (2003).
[33] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, Hybrid metaheuristics in combinatorial optimization: A survey, Applied soft computing **11**, 4135 (2011).
[34] G. Oliveri and J. T. Overvelde, Inverse design of mechanical metamaterials that undergo buckling, Adv. Funct. Mater. **30**, 1909033 (2020).
[35] Y. Bengio, A. Lodi, and A. Prouvost, Machine learning for combinatorial optimization: A methodological tour d'horizon, Eur. J. Oper. Res. **290**, 405 (2021).
[36] E. Pitzer and M. Affenzeller, A comprehensive survey on fitness landscape analysis, *Recent Advances in Intelligent Engineering Systems*, Vol. 378 (Springer, Berlin, Heidelberg, 2012), pp. 161–191.
[37] K. Bertoldi, V. Vitelli, J. Christensen, and M. Van Hecke, Flexible mechanical metamaterials, Nat. Rev. Mater. **2**, 17066 (2017).
[38] K. E. Evans and A. Alderson, Auxetic materials: Functional materials and structures from lateral thinking! Adv. Mater. **12**, 617 (2000).
[39] J. N. Grima, A. Alderson, and K. Evans, Auxetic behaviour from rotating rigid units, Physica Status Solidi (b) **242**, 561 (2005).
[40] D. Acuna, F. Gutiérrez, R. Silva, H. Palza, A. S. Nunez, and G. Düring, A three step recipe for designing auxetic materials on demand, Commun. Phys. **5**, 113 (2022).
[41] M. A. Bessa, P. Glowacki, and M. Houlder, Bayesian machine learning in metamaterial design: Fragile becomes supercompressible, Adv. Mater. **31**, 1904845 (2019).

[42] A. J. Lew, K. Jin, and M. J. Buehler, Designing architected materials for mechanical compression via simulation, deep learning, and experimentation, npj Comput. Mater. **9**, 80 (2023).

[43] J. Berger, H. Wadley, and R. McMeeking, Mechanical metamaterials at the theoretical limit of isotropic elastic stiffness, Nature (London) **543**, 533 (2017).

[44] W. Deng, S. Kumar, A. Vallone, D. M. Kochmann, and J. R. Greer, Ai-enabled materials design of non-periodic 3d architectures with predictable direction-dependent elastic properties, Adv. Mater. **36**, 2308149 (2024).

[45] D. Z. Rocklin, S. Zhou, K. Sun, and X. Mao, Transformable topological mechanical metamaterials, Nat. Commun. **8**, 14201 (2017).

[46] J. Z. Kim, Z. Lu, S. H. Strogatz, and D. S. Bassett, Conformational control of mechanical networks, Nat. Phys. **15**, 714 (2019).

[47] A. Bossart, D. M. Dykstra, J. van der Laan, and C. Coulais, Oligomodal metamaterials with multifunctional mechanics, Proc. Natl. Acad. Sci. USA **118**, e2018610118 (2021).

[48] Z. Meng, M. Liu, H. Yan, G. M. Genin, and C. Q. Chen, Deployable mechanical metamaterials with multistep programmable transformation, Sci. Adv. **8**, eabn5460 (2022).

[49] R. van Mastrigt, C. Coulais, and M. van Hecke, Emergent nonlocal combinatorial design rules for multimodal metamaterials, Phys. Rev. E **108**, 065002 (2023).

[50] D. M. J. Dykstra and C. Coulais, Inverse design of multishape metamaterials, Phys. Rev. Appl. **22**, 064013 (2024).

[51] A. Bossart and R. Fleury, Extreme spatial dispersion in nonlocally resonant elastic metamaterials, Phys. Rev. Lett. **130**, 207201 (2023).

[52] Y. Cho, J.-H. Shin, A. Costa, T. A. Kim, V. Kunin, J. Li, S. Y. Lee, S. Yang, H. N. Han, I.-S. Choi *et al.*, Engineering the shape and structure of materials by fractal cut, Proc. Natl. Acad. Sci. USA **111**, 17390 (2014).

[53] D. M. Sussman, Y. Cho, T. Castle, X. Gong, E. Jung, S. Yang, and R. D. Kamien, Algorithmic lattice kirigami: A route to pluripotent materials, Proc. Natl. Acad. Sci. USA **112**, 7449 (2015).

[54] J. T. Overvelde, T. A. De Jong, Y. Shevchenko, S. A. Becerra, G. M. Whitesides, J. C. Weaver, C. Hoberman, and K. Bertoldi, A three-dimensional actuated origami-inspired transformable metamaterial with multiple degrees of freedom, Nat. Commun. **7**, 10929 (2016).

[55] J. T. Overvelde, J. C. Weaver, C. Hoberman, and K. Bertoldi, Rational design of reconfigurable prismatic architected materials, Nature (London) **541**, 347 (2017).

[56] P. Dieleman, N. Vasmel, S. Waitukaitis, and M. van Hecke, Jigsaw puzzle design of pluripotent origami, Nat. Phys. **16**, 63 (2020).

[57] J. McInerney, B. G.-g. Chen, L. Theran, C. D. Santangelo, and D. Z. Rocklin, Hidden symmetries generate rigid folding mechanisms in periodic origami, Proc. Natl. Acad. Sci. USA **117**, 30252 (2020).

[58] D. Melancon, A. E. Forte, L. M. Kamp, B. Gorissen, and K. Bertoldi, Inflatable origami: Multimodal deformation via multistability, Adv. Funct. Mater. **32**, 2201891 (2022).

[59] B. Aksoy and H. Shea, Multistable shape programming of variable-stiffness electromagnetic devices, Sci. Adv. **8**, eabk0543 (2022).

[60] W. Liu, S. Janbaz, D. Dykstra, B. Ennis, and C. Coulais, Harnessing plasticity in sequential metamaterials for ideal shock absorption, Nature **634**, 842 (2024).

[61] R. van Mastrigt, M. Dijkstra, M. van Hecke, and C. Coulais, Machine learning of implicit combinatorial rules in mechanical metamaterials, Phys. Rev. Lett. **129**, 198003 (2022).

[62] B. Pisanty, E. C. Oğuz, C. Nisoli, and Y. Shokef, Putting a spin on metamaterials: Mechanical incompatibility as magnetic frustration, SciPost Phys. **10**, 136 (2021).

[63] K. Sedlaczek and P. Eberhard, Topology optimization of large motion rigid body mechanisms with nonlinear kinematics, J. Comput. Nonlinear Dyn. **4**, 021011 (2009).

[64] L. Zimmermann, K. Shea, and T. Stanković, A computational design synthesis method for the generation of rigid origami crease patterns, J. Mech. Rob. **14**, 031014 (2021).

[65] J. Wang, J. Callanan, O. Ogunbodede, and R. Rai, Hierarchical combinatorial design and optimization of non-periodic metamaterial structures, Addit. Manuf. **37**, 101710 (2021).

[66] T. S. Lumpe and K. Shea, Computational design of multi-state lattice structures with finite mechanisms for shape morphing, J. Mech. Des. **145**, 071701 (2023).

[67] S. Van't Sant, P. Thakolkaran, J. Martínez, and S. Kumar, Inverse-designed growth-based cellular metamaterials, Mech. Mater. **182**, 104668 (2023).

[68] S. Liu and P. Acar, Parameter space exploration of cellular mechanical metamaterials using genetic algorithms, AIAA J. **61**, 3633 (2023).

[69] S. L. Lovelock, R. Crawshaw, S. Basler, C. Levy, D. Baker, D. Hilvert, and A. P. Green, The road to fully programmable protein catalysis, Nature (London) **606**, 49 (2022).

[70] Z. Zeravcic, V. N. Manoharan, and M. P. Brenner, Size limits of self-assembled colloidal structures made using specific interactions, Proc. Natl. Acad. Sci. USA **111**, 15918 (2014).

[71] C. G. Evans, J. O'Brien, E. Winfree, and A. Murugan, Pattern recognition in the nucleation kinetics of non-equilibrium self-assembly, Nature (London) **625**, 500 (2024).

[72] P. Merrell, Example-based model synthesis, *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (Association for Computing Machinery, New York, NY, 2007), pp. 105–112.

[73] P. Merrell and D. Manocha, Model synthesis: A general procedural modeling algorithm, IEEE Trans. Vis. Comput. Graph. **17**, 715 (2010).

[74] C. Bilodeau, W. Jin, T. Jaakkola, R. Barzilay, and K. F. Jensen, Generative models for molecular discovery: Recent advances and challenges, WIREs Comput. Mol. Sci. **12**, e1608 (2022).

[75] A. S. Meeussen, E. C. Oğuz, Y. Shokef, and M. van Hecke, Topological defects produce exotic mechanics in complex metamaterials, Nat. Phys. **16**, 307 (2020).

[76] R. D. Resch, Geometrical device having articulated relatively movable sections (1965), United States of America Patent 3201894.

[77] T. Mullin, S. Deschanel, K. Bertoldi, and M. C. Boyce, Pattern transformation triggered by deformation, Phys. Rev. Lett. **99**, 084301 (2007).

[78] K. Bertoldi, P. M. Reis, S. Willshaw, and T. Mullin, Negative poisson's ratio behavior induced by an elastic instability, Adv. Mater. **22**, 361 (2010).

[79] C. Coulais, J. T. Overvelde, L. A. Lubbers, K. Bertoldi, and M. van Hecke, Discontinuous buckling of wide beams and metabeams, Phys. Rev. Lett. **115**, 044301 (2015).

[80] C. Coulais, C. Kettenis, and M. van Hecke, A characteristic length scale causes anomalous size effects and boundary programmability in mechanical metamaterials, Nat. Phys. **14**, 40 (2018).

[81] M. Czajkowski, C. Coulais, M. van Hecke, and D. Z. Rocklin, Conformal elasticity of mechanism-based metamaterials, Nat. Commun. **13**, 211 (2022).

[82] Y. P. Hong and C.-T. Pan, Rank-revealing *QR* factorizations and the singular value decomposition, Math. Comp. **58**, 213 (1992).

[83] M. Gu and S. C. Eisenstat, Efficient algorithms for computing a strong rank-revealing *QR* factorization, SIAM J. Sci. Comput. **17**, 848 (1996).

[84] L. Wu, L. Liu, Y. Wang, Z. Zhai, H. Zhuang, D. Krishnaraju, Q. Wang, and H. Jiang, A machine learning-based method to design modular metamaterials, Extreme Mech. Lett. **36**, 100657 (2020).

[85] F. Dos Reis and N. Karathanasopoulos, Inverse metamaterial design combining genetic algorithms with asymptotic homogenization schemes, Int. J. Solids Struct. **250**, 111702 (2022).

[86] We estimate this fraction by extrapolation of the exponential decay shown in Fig. [2(a)].

[87] We neglect the trivial global CRS mode that is always present, as we can remove this mode while retaining all other modes by adding a single extra horizontal or vertical rigid bar to the design.

[88] C. Bron and J. Kerbosch, Algorithm 457: Finding all cliques of an undirected graph, Commun. ACM **16**, 575 (1973).

[89] S. K. Patiballa and G. Krishnan, On the design of three-dimensional mechanical metamaterials using load flow visualization, Mech. Based Des. Struct. Mach. **50**, 442 (2022).

[90] Q. Zeng, S. Duan, Z. Zhao, P. Wang, and H. Lei, Inverse design of energy-absorbing metamaterials by topology optimization, Adv. Sci. **10**, 2204977 (2023).

[91] S. Hormoz and M. P. Brenner, Design principles for self-assembly with short-range interactions, Proc. Natl. Acad. Sci. USA **108**, 5193 (2011).

[92] F. M. Gartner, I. R. Graf, and E. Frey, The time complexity of self-assembly, Proc. Natl. Acad. Sci. USA **119**, e2116373119 (2022).

[93] M. van Hecke, Profusion of transition pathways for interacting hysterons, Phys. Rev. E **104**, 054608 (2021).

[94] H. Bense and M. van Hecke, Complex pathways and memory in compressed corrugated sheets, Proc. Natl. Acad. Sci. USA **118**, e2111436118 (2021).

[95] D. Shohat, D. Hexner, and Y. Lahini, Memory from coupled instabilities in unfolded crumpled sheets, Proc. Natl. Acad. Sci. USA **119**, e2200028119 (2022).

[96] R. van Mastrigt, Code for calculating zero modes, https://uva-hva.gitlab.host/published-projects/CombiMetaMaterial (2023).

[97] R. van Mastrigt, Code for inverse design, https://uva-hva.gitlab.host/published-projects/inversecombimat (2024).

[98] R. van Mastrigt, M. Dijkstra, M. van Hecke, and C. Coulais, "Computational design of multimodal combinatorial mechanical metamaterials (1.0.0)" [Data set], Zenodo (2024), https://doi.org/10.5281/zenodo.12190638.

[99] J. C. Maxwell, L. on the calculation of the equilibrium and stiffness of frames, London, Edinburgh Dublin Philos. Mag. J. Sci. **27**, 294 (1864).

[100] C. Calladine, Buckminster fuller's "tensegrity" structures and clerk maxwell's rules for the construction of stiff frames, Int. J. Solids Struct. **14**, 161 (1978).

[101] T. Lubensky, C. Kane, X. Mao, A. Souslov, and K. Sun, Phonons and elasticity in critically coordinated lattices, Rep. Prog. Phys. **78**, 073901 (2015).

[102] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry *et al.*, Sympy: Symbolic computing in python, PeerJ Comput. Sci. **3**, e103 (2017).

[103] L. Perron and F. Didier, Cp-sat, https://developers.google.com/optimization/cp/cp_solver/ (2023).

[104] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX: Composable transformations of Python+NumPy programs, http://github.com/google/jax (2018).

[105] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, May 7–9, 2015*, edited by Y. Bengio and Y. LeCun, Conference Track Proceedings (ICLR, 2015).

[106] D. Eppstein, M. Löffler, and D. Strash, Listing all maximal cliques in sparse graphs in near-optimal time, in *Algorithms and Computation: ISAAC 2010*, edited by O. Cheong, K. Y. Chwa, and K. Park, Lecture Notes in Computer Science, Vol. 6506 (Springer, Berlin, Heidelberg, 2010), pp. 403–414.