Mapping the discrete folding landscape

João C. Neves[1,2*], Bernardo R. Marques[1,2*], Cristóvão S. Dias[1,2†] and Nuno A. M. Araújo[1,2†]

[1*]Departamento de Física, Faculdade de Ciências, Universidade de Lisboa, Lisboa, 1749-016, Portugal.
[2]Centro de Física Teórica e Computacional, Universidade de Lisboa, Lisboa, 1749-016, Portugal.

*Corresponding author(s). E-mail(s): jlneves@fc.ul.pt; fc59924@alunos.ciencias.ulisboa.pt;
Contributing authors: csdias@fc.ul.pt; nmaraujo@fc.ul.pt;
†These authors contributed equally to this work.

Abstract

Folding is emerging as a promising manufacturing process to transform flat materials into functional structures, offering efficiency by reducing the need for welding, gluing, and molding, while minimizing waste and enabling automation. Designing target shapes requires not only to determine cuts and folds, but also folding pathways. Simple combinatorics is impractical as the possibilities grow factorially with the number of folds. To address this, we present a graph-based algorithm for polyhedral shapes. By representing the target shape as a graph, where nodes correspond to faces and edges represent adjacency, the algorithm identifies all possible fold sequences and maps the configuration space into a discrete set of intermediate configurations. This systematic mapping is critical for the design of optimized processes, the simplifying of folding operations, the reduction of failures, and the improvement of manufacturing reliability.

Keywords: Folding, graph theory, algorithm

1 Introduction

Folding is becoming increasingly relevant in modern manufacturing due to its remarkable versatility [1, 2]. By enabling intricate 3D shapes from flat materials while reducing the need for adhesives and joints, folding significantly minimizes material waste compared to conventional methods [3]. It also offers exceptional flexibility, supports automation, and facilitates rapid prototyping through laser cutting and 3D printing technologies. These benefits make folding a transformative tool in industries such as robotics, packaging, aerospace, and biomedical engineering [4–12].

The idea of using folding to build structures applies across various length scales (see Fig. 1), from nanoscale bioengineered DNA networks [13, 14] to macroscale foldable satellites [4]. Microscale applications include cell-activated hinges [15], while centimeter-scale innovations feature self-folding robots [4]. Additional applications span soft robotics [4, 5], biomedical devices [5, 10, 11], space technology [6, 7], and drug delivery systems [15, 16]. Folding methods have also been used to produce meta-materials using strategies such as creating a well-defined pattern of folds like in the Miura-Ori [17–19], cuts [20] or by creating foldable sub-structures which are modular and can be combined to produce more complex shapes [21]. Foldable meta-materials provide functions ranging from ultra-stiffness [22]

and negative Poisson ratios [17] to negative thermal expansion [23], impact absorption [24–26], and the development of programmable [27, 28] or tunable shapes [29]. While folding mechanisms and binding strategies vary across scales and systems, some features, such as the set of intermediate configurations, depend solely on geometry and topology, making them independent of scale or material.

The concept of folding 3D shapes from flat (2D) templates originates from the ancient art of origami, in which paper (*gami*) is folded into 3D shapes through precise sequences of folds (*ori*). The introduction of cuts (*kiri*), as in kirigami, expands this capability to more intricate shapes. A classic example is the Latin cross template, consisting of a row of four adjacent squares with additional squares on each side of the second square, which forms a cube when folded along its edges (see Fig. 2.A). Such templates, also known as nets, represent the 2D unfolding of 3D structures and were first formalized in the 16th century by Albrecht Dürer [30].

Significant research has focused on identifying templates for specific folded structures and predicting the resulting 3D shapes. For polyhedral shapes, templates can be obtained by edge unfolding, where a set of cuts along edges allows the structure to flatten into a single piece [31]. However, these cuts must respect certain constraints: they must not form a loop to prevent multiple disconnected pieces, and they must visit all vertices to ensure a flat unfolded structure. Determining intermediate configurations and folding pathways, however, remains challenging. If we assume that all configurations with the same set of connected faces (uncut edges) are equivalent, regardless of the angles between faces along the free-moving edges, then the configuration landscape becomes discrete. Once the required edges for unfolding are known, identifying intermediate structures reduces to evaluating all possible subsets of these cuts. However, the number of possibilities grows factorially with the number of edges, making the task impractical. Even for a cube, which requires cutting along only seven edges, there are 5,040 possible sequences to be tested. For an icosahedron, with just four more cuts, the number of sequences explodes to 39 million. In addition, cutting along an edge produces a new intermediate configuration but does not necessarily lead to a new folding state, as the number of free-moving edges remains the same. For example, in a cube, at least three edges must be cut along before a face can unfold, an aspect that straightforward combinatorial analysis fails to capture. Alternatively, direct simulations of foldings can be attempted, but these methods often fail to account for kinetically improbable structures, leaving many configurations unexplored [32]. The current methods for probing the configurational space rely on searching for vertex connections on templates and sequentially connect them, leading to intermediate structures and therefore pathways of folding. This approach, however selects only a subset of intermediate structures which despite being kinetically more probable do not correspond to the whole set of intermediate structures [33, 34]. Our method is able to completely map the configurational space given an initial and target structure while maintatining its efficiency much below the factorial explosion (See Supplementary Material D).

It has been shown that both shapes and templates can be mapped onto graphs [31, 32, 35]. One graph is the *face graph*, where the nodes are the faces and links indicate adjacency. The other is the *shell graph*, where the nodes correspond to the vertices of the polyhedron, and links represent its edges. Since different structures yield different graph representations, adding or removing links in a graph corresponds to folding or unfolding faces. Thus, the face graphs of the shape and of the corresponding templates share the same nodes and differ only in the number of links. Within the graph representation framework, templates can be uniquely and deterministically obtained by identifying spanning trees in the *face graph* of the shape [35]. To identify intermediate configurations, we first determine the set of cuts that lead to unfolding while accounting for previously visited configurations, reducing the number of evaluations needed. Our algorithm, given any initial and final structure, searches for all possible folding paths and intermediate configurations. This approach allows us to construct a complete catalog of all possible intermediate configurations and to map all paths between them, thereby representing the discrete folding landscape as a directed graph.

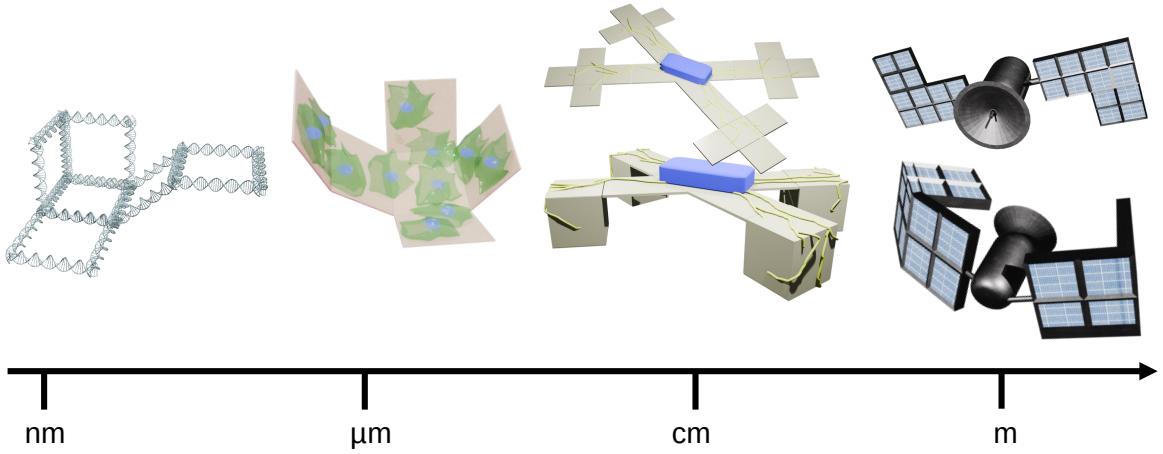

Fig. 1 Examples of foldable structures at different scales. From DNA strands origami structures at the nanometer scale [13, 14], to cell-activated hinges [15], to self-folding robots [4] and to large-scale foldable telescopes.

2 Model Description

To describe the proposed algorithm, consider the case of a cube and its corresponding Latin cross template (Fig. 2.A). The associated *shell* and *face* graphs are shown in Fig. 2.B. Unfolding the cube by cutting along its edges corresponds to removing links in the face graph. The set of all removed links (or edges) forms the cut graph, a subgraph of the face or shell graph that maintains the same nodes but with fewer links. Specifically, the cut graph of the face graph represents the difference between the face graphs of the folded shape and of its unfolded template, while the cut graph of the shell graph corresponds to the set of edges that must be cut along on the cube to fully unfold it (see Fig. 2.0).

Once we define the initial and final face graphs, to identify all intermediate folding states, we must determine all possible combinations of link removals (edge cuts) that produce valid unfolding steps. However, removing a single link does not always allow a face to unfold. As an example, Fig. 3.A shows the 24 configurations obtained by cutting edges of the cube (marked in red) that do not lead to a new folding state, as no face can be unfolded. To systematically identify all intermediate folding states, we begin by choosing a random link from the cut graph and removing it (cutting along the edge) (Fig. 2.1). All adjacent links on the shell cut graph (those sharing a node with the removed link) are then added to a list of candidates for the next removal (Fig. 2.2), and the resulting graph is stored as a child configuration on the relational map (Fig. 2.3). This new configuration is not necessarily a new folding state. Nevertheless, we retain all configurations to reconstruct the full folding sequence and ensure a complete pathway analysis.

After steps (1–3), we check whether the current configuration has been visited before. To do this efficiently, we encode each configuration as a binary number, where each digit corresponds to a specific link in the cut graph: set to zero if the link has been removed and one otherwise. This binary representation uniquely identifies each configuration (see examples in *Supporting Information* B). If the configuration is new, we determine whether it represents a distinct intermediate folding state, meaning that at least one additional face can unfold compared to its parent configuration. If not, it is the same folding state. The edges along which unfolding can occur correspond to specific links in the face graph (known as red links [36, 37]) that, when removed, cause the graph to split into two disconnected components (see *Supporting Information* A). Thus, a new folding state is characterized by an increase in the number of these foldable edges relative to the previous one. Figure 3 shows all folding states and corresponding configurations for (a) zero, (b) one, (c) two, (d) three, and (e)

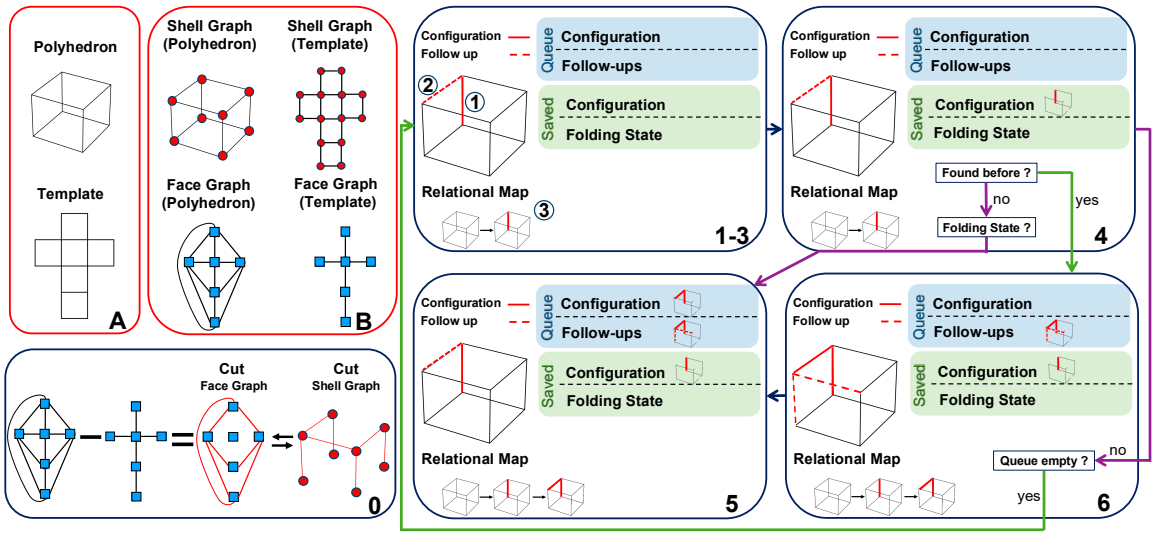

Fig. 2 Scheme of the algorithm. The folded polyhedron (cube) and its unfolded template (Latin cross) **(A)** serve as the starting point of the algorithm. Based on these structures, we construct the shell graph and the face graphs of both the template and the folded polyhedron **(B)**. These three structures act as inputs to the algorithm. The process begins by constructing the face graph of the cut, obtained by subtracting the face graph of the template from the face graph of the folded polyhedron. This step also yields the shell graph of the cut **(0)**. The algorithm selects an initial link (edge) on the shell graph, defining the first configuration **(1)**. Adjacent links to this starting link are identified as potential follow-ups **(2)**. The folded polyhedron is assigned as the parent configuration of the starting-link configuration **(3)**. The algorithm checks whether this configuration has been encountered before. If it has, the algorithm moves on to **6**; else, it is saved as a new configuration **(4)**. If the configuration is new, the next configurations to evaluate are generated by adding the identified follow up links to the current configuration. These configurations, along with their respective follow-ups, are added to a queue. Each new configuration is designated as a child of the current one in the relational map **(5)**. A new configuration is retrieved from the queue, and the process returns to step **(4)**. When the queue is exhausted, the algorithm loops back to **(1)** and selects a new starting link. If no new links remain and the queue is empty, the algorithm terminates.

five foldable edges. As shown in Fig. 2.4, newly identified configurations are added to a list, and the process advances to step 5; otherwise, it proceeds to step 6.

In step 5 (see Fig. 2.5), new configurations are generated by selecting the next adjacent link from the cut graph, based on the list obtained in step 2. The number of possible new configurations corresponds to the number of adjacent links. These new configurations are added to a queue, each paired with its corresponding list of adjacent links associated with the newly selected link (as in step 2). These configurations are labeled as children of the previous one in the relational map, where each connection defines a path that only needs to be explored once, significantly improving the efficiency of the algorithm. In step 6 (Fig. 2.6), we iterate through every configuration in the queue and repeat the process starting from step 4. Once the queue is empty, we return to step 1 (Fig. 2.1) and choose a different starting edge, continuing until all edges are selected.

So far, we have only identified sets of contiguous removed links, leading to pathways that require a connected path of cut edges in the shell graph (blue states in Fig. 3.C). However, a shape can also be unfolded by cutting along in different, non-adjacent regions (green states in Fig. 3.C). These pathways can be obtained by applying the algorithm to the newly found folding states with each folding state being considered as a targeted shape for the algorithm. New folding states found by recursively applying the algorithm will be unfolded in the same way. When all folding states have been identified, the algorithm stops.

The result is a set of configurations, corresponding to subgraphs of the shell graph, grouped into intermediate folding states. The relational map is a directed network of different configurations, with the folded shape on one end and the unfolded template on the other. If we only consider the folding configurations, the network corresponds to the discrete folding landscape, where each node represents

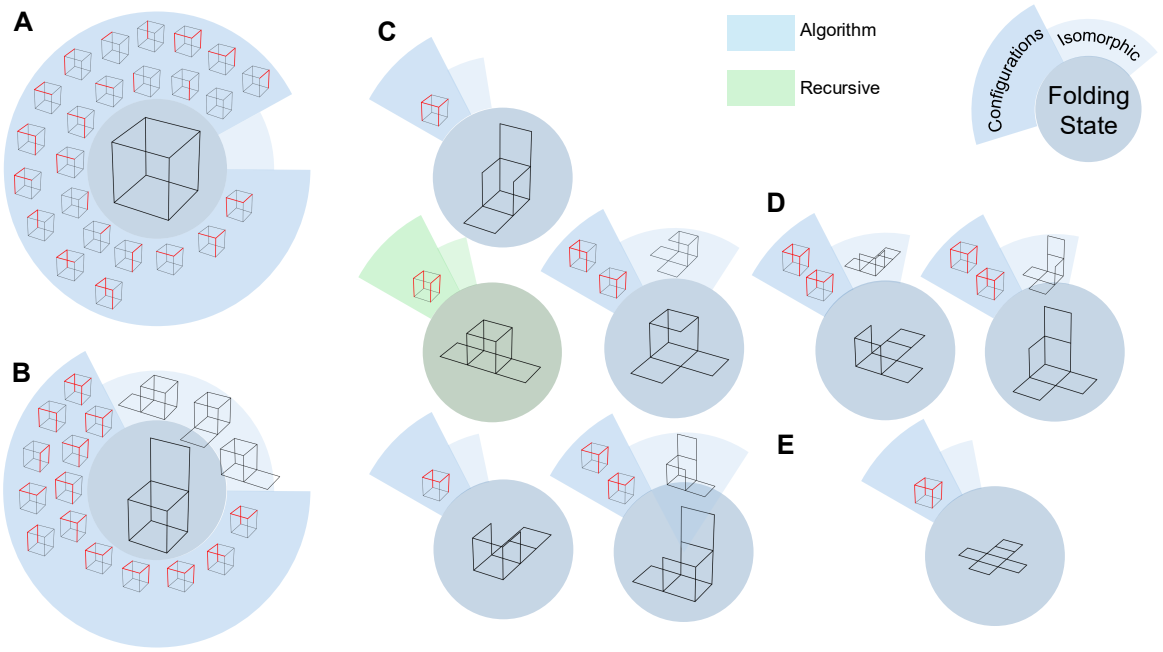

Fig. 3 Catalog of folding states and configurations for the cube and Latin cross. Two configurations differ in the set of cut edges, while folding states differ in the set of foldable edges. **(A)** There are 24 configurations of the cube where no face can be unfolded (i.e., zero foldable edges), despite having certain edges cut (marked in red). **(B)** When one foldable edge is present, there are 14 distinct configurations, corresponding to four folding states. However, these four folding states are isomorphic, so they are grouped into a single folding state. **(C)–(E)** Scheme of the folding states, their isomorphisms, and the corresponding configurations for cases with two, three, and five foldable edges. Notably, no folding state exists with exactly four foldable edges. In **(C)** we distinguish between folding states obtained through a connected path of cut edges (in blue) and those arising from a combination of disconnected cuts (in green), these folding states are obtained by recursively applying the algorithm.

a folding state, and the links between nodes indicate possible connections through a sequence of edge cuts. The connection between folding configurations is obtained by going backwards and grouping configurations that do not lead to a new folding state.

At the end, several intermediate states are isomorphic to each other. For example, in the case of the Latin cross, there are four identical folding configurations where all but one face is folded (Fig. 3). We group all isomorphic configurations into one, simplifying the discrete landscape. To detect isomorphic states, we analyze the face graphs of different configurations and apply the VF2++ algorithm [38], implemented in the NetworkX package [39]. Note that different templates may share the same face graph; therefore, to distinguish them, we must also check for geometric differences. Once we have the entire landscape for a certain template, we can incorporate all the other known templates to obtain the full discrete folding landscape. For instance, in the case of the cube, in addition to the Latin cross, there are ten other possible templates. The folding landscape for the cube is shown in Fig. 4, with the paths corresponding to the Latin cross highlighted in red. Notably, several folding states are common across different templates. Thus, when searching for intermediate configurations and folding states, it is computationally efficient to also consider those obtained from other templates. In the *Supporting Information D*, we demonstrate that for very large shapes, the number of newly discovered configurations and states increases sublinearly with the number of added templates.

3 Results and Discussion

Figure 4 shows the folding landscape for the cube with its eleven templates and eighteen intermediate folding states. This network includes all possible non-isomorphic folding states (nodes) and the transitions between them via a sequence of edge cuts (links). The algorithm also retrieves the configurations corresponding to each state (see Fig. 3) and the sequence of edge cuts for each link, providing a detailed description of the folding process.

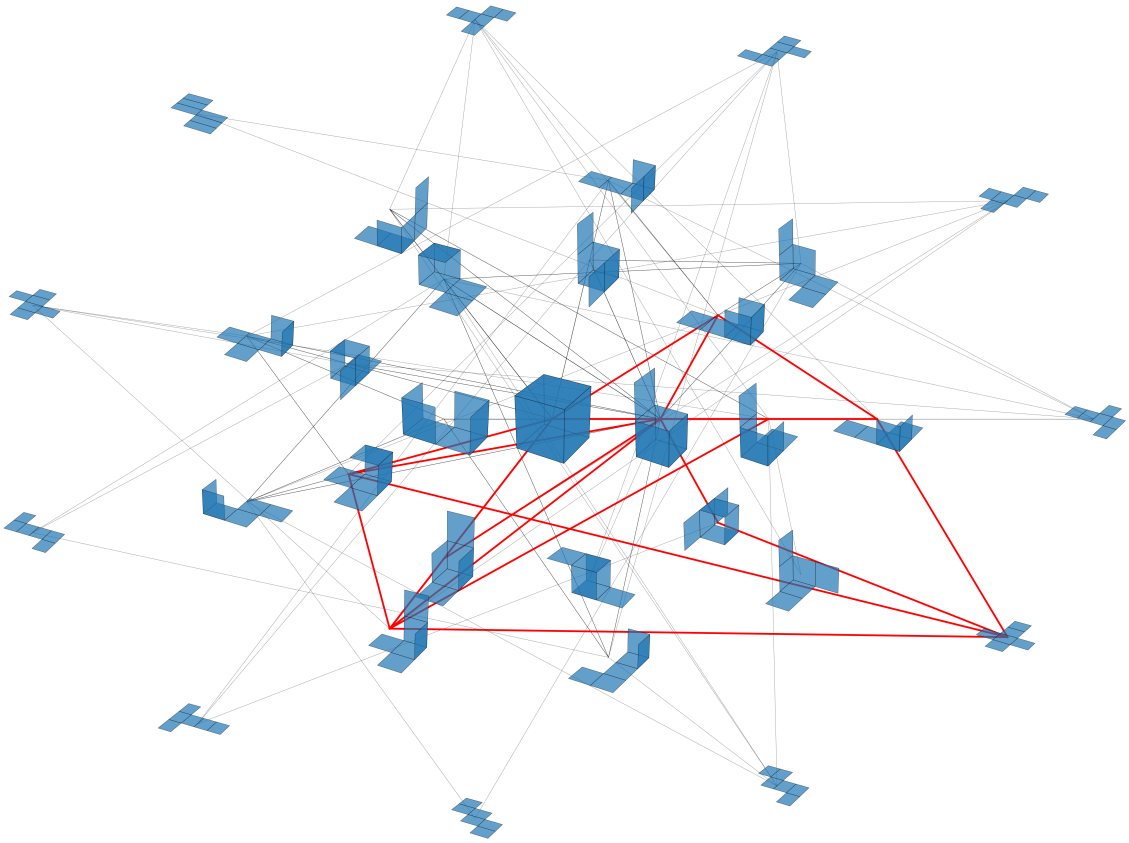

Fig. 4 Folding landscape for the cube. The algorithm retrieves a network where each node represents a non-isomorphic folding state, and the links indicate possible transitions between states through a sequence of edge cuts. The cube has eleven distinct templates. The structure in the center (cube) has no red links and the templates have five, structures with the same number of red links are organized in concentric circles with ascending order by their number of red links. The folding pathways for the Latin cross are highlighted in red.

The analysis of this landscape reveals all possible pathways and connections between the folding states. Each template offers multiple pathways to reach the fully folded cube, with the relative statistical weight depending on the folding mechanism. At the microscale, folding can be driven by thermal motion, where the sequence of folds is determined by the rate at which two faces bind to each other [40, 41]. At the macroscale, folding is typically controlled by actuators or stress relaxation [4, 27], allowing pathways to be designed either to minimize the number of folds or to reduce the use of actuators.

Knowing the folding landscape is also essential for the design of multifarious templates, which can fold into multiple structures [42]. These templates hold promise for the development of shape-changing materials capable of adapting their form in response to external stimuli. Obtaining their folding landscape is straightforward; we apply the algorithm to all possible closed structures. Figure 5.A (blue structures) presents an example of a template that can fold into either an octahedron or a tritetrahedral (boat) configuration. Identifying possible pathways and critical folds that lead to different final structures can inform the design of the folding process and provide insight into how the final configuration depends on temperature in thermally driven folding [43].

Up to this point, we have assumed that the final folded state is always a closed polyhedron. However, some folded states may correspond to misfolds, which are partially folded structures that cannot fully assemble into the intended polyhedron without additional edge cuts. These misfolds can be identified through tedious combinatorial analysis or detected dynamically in direct simulations or experiments. Once identified, they must be incorporated into the algorithm as inputs to construct a more extensive folding landscape (Fig. 5). By doing so, we obtain a comprehensive representation of

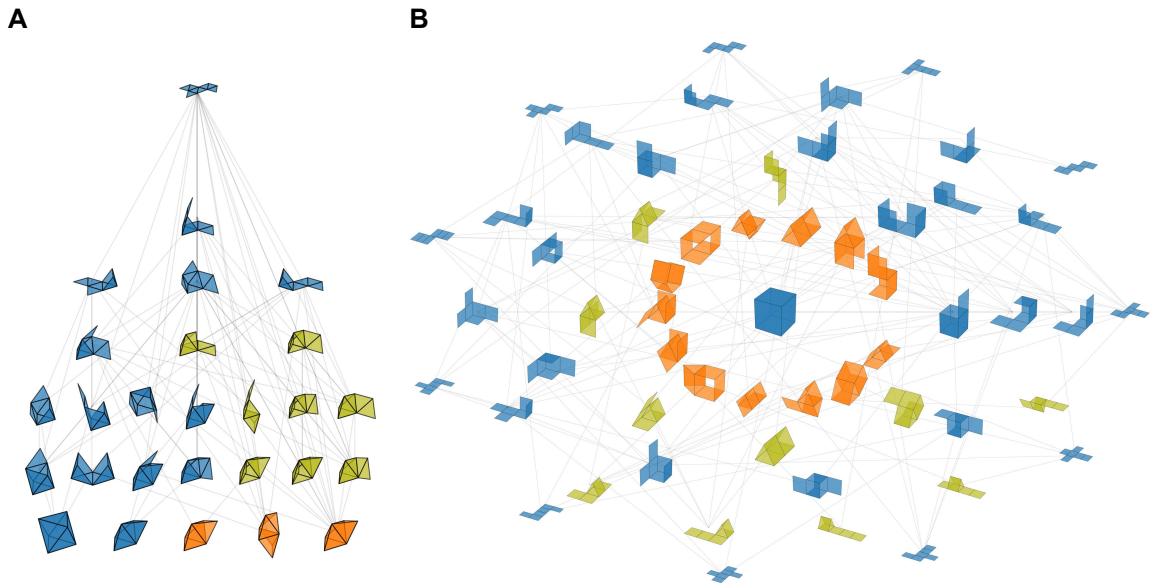

Fig. 5 Folding landscape for the cube and a multifarious template with misfolds. Similarly to Figures 4 the network of templates, target structures and intermediate folding states is here shown with additional misfolded structures (in yellow and orange) as well. On **A** we analyze a multifarious template capable of folding into either an octahedron or a tritetrahedral (boat) structure. A multifarious template is one that can fold into more than one closed structure. To construct the folding landscape, we apply the proposed algorithm, starting from both closed configurations of the same template. Structures are organized by their number of red links starting from the top (the template with seven) to the bottom (the target structures with zero). To the octahedron and the tritetrahedral (boat) structure we add three additional misfolds to which the template can evolve to. These structures in orange with no red links were added as the target structure input to the algorithm together with their respective template. Intermediate folding states which only evolve towards misfolds are shown in yellow. On **B** several templates and the target structure, the cube, are joined by misfolded folding states added as target structures to the algorithm (orange structures).

the folding landscape, including misfolded states, as shown in *Supplementary Material* C for the two cases considered here: the cube and the multifarious template.

4 Conclusion

We introduced a graph-based algorithm for mapping the complete discrete folding landscape of three-dimensional (3D) structures from two-dimensional (2D) templates consisting of panels connected by flexible hinges. By representing polyhedral templates as graphs of connected faces and edges, our approach efficiently identifies all possible intermediate folding configurations, folding states, and transition pathways. This method overcomes the factorial growth limitations of traditional combinatorial approaches, providing a computationally feasible solution for analyzing complex folding processes.

A key advantage of this framework is its ability to capture not only valid folding sequences but also misfolds, which are states that cannot fully assemble into the intended structure without additional edge cuts. By incorporating these misfolds into the analysis, the algorithm provides a comprehensive representation of the folding landscape, including alternative pathways to correct folding errors. The approach also extends to multifarious templates, which can be folded into multiple distinct closed structures.

The insights gained from this folding landscape have direct implications for self-folding materials, origami-based robotics, and deployable structures. At the microscale, where folding is driven by thermal motion, understanding the sequence of binding interactions can help optimize assembly kinetics [12, 14, 32, 40]. At the macroscale, where actuators and mechanical constraints govern folding, the landscape provides a means to design efficient folding pathways that minimize energy expenditure or mechanical complexity [4]. The type and length scale of the system will have direct consequences on its dynamics, these specific details are captured in the distribution of weights for links in the obtained

network of folding. However, the network itself is not dependent on the scale and can be studied solely from an abstract point of view.

Beyond its immediate applications, our method lays the foundation for machine learning-driven folding design. The dataset generated from our algorithm comprises folding pathways, misfolded states, and transition probabilities, that can be used to train models that predict novel and optimal folding strategies. This has potential applications in the development of programmable meta-materials, reconfigurable devices, and adaptive structures.

Finally, we demonstrate that as more templates are added, the number of newly discovered configurations and folding states grows sublinearly, suggesting that our approach remains computationally efficient even for large and complex structures. Future work will explore how external forces, geometric constraints, and stochastic effects influence folding landscapes, further refining our understanding of how foldable structures can be designed, controlled, and optimized across scales and disciplines.

Supplementary information. The code for the algorithm is available on https://github.com/jneves-cftc/kirigami.

Acknowledgements. We acknowledge financial support from the Portuguese Foundation for Science and Technology (FCT) under the contracts: UIDB/00618/2020 (DOI:10.54499/UIDB/00618/2020), UIDP/00618/2020 (DOI:10.54499/UIDP/00618/2020), and 2023.00707.BD.

Appendix A Red links

In graph theory, a red link is a link that, when removed separates the structure into two distinct components. The structure on Fig. A1.A has five red links whereas structure A1.B has none. If one link from the face graph of structure A1.A is removed, i.e. to cut along any edge in the template, the structure splits into two distinct components, as shown in Fig. A1.C. On the other hand, cuts along any edge of the cube do not split the structure.

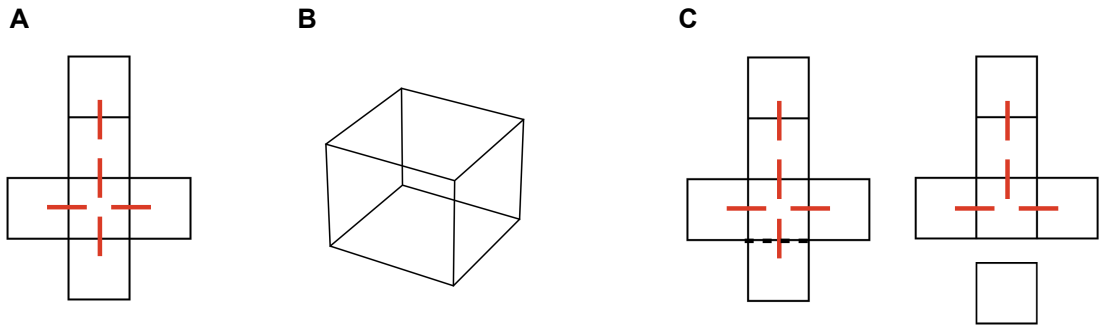

Fig. A1 Red links on the template and folded structure. Red links are used to determine which configurations correspond to folding states. The latin cross template of the cube **A** has 5 red links, these correspond to links on the face graph which when cut separate the graph into two unconnected components as shown on **C**. The cube **B** on the other hand has 0 red links, there is no unique edge which when cut separates this structure into two.

The number of red links in a specific structure is obtained by applying a burning algorithm [37]. In Fig. A2 we show the employment of this algorithm on the two structures of Fig. A1.C. The algorithm starts by choosing a node to *burn*, as we are dealing with the face graph of the structure, nodes correspond to faces. On the following iterations, the fire spreads to the neighbors of the burnt nodes, that is, their adjacent faces. The process ends when there are no neighbors left to burn. In the end if all nodes have been burned, the structure is still one piece, if not, the structure is made of two separate components.

To know if a certain link is a red link, we remove it from the face graph and apply this algorithm. If after the link is removed the structure is still one piece, the link is not a red link. If the structure is made of two separate components, the removed link is a red link.

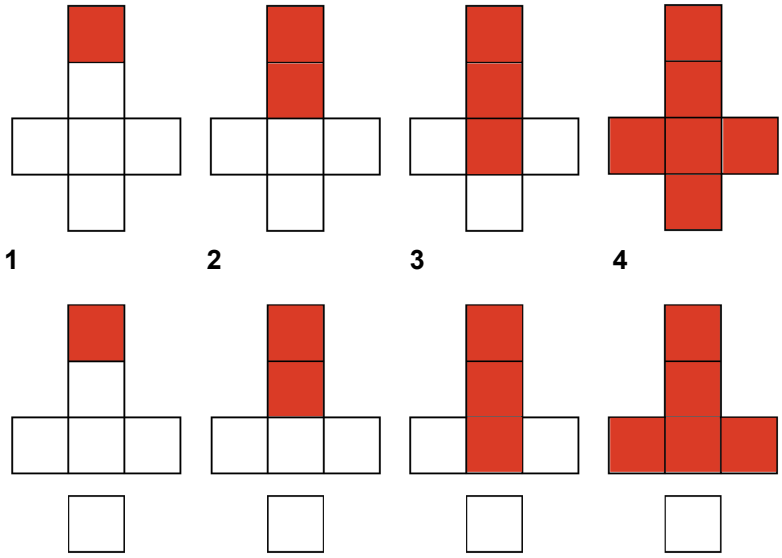

Fig. A2 Burning algorithm. Red links are found using a burning algorithm. The algorithm starts by choosing a node to *burn* **1**, on the following iterations the neighbors of the burnt node will also be burnt **2**, **3**, **4**. The algorithm ends when there are no neighbors left to burn. The structure on top is one piece, thus its nodes (or faces) are all burned in the end. The structure on the bottom corresponds to two separate components, so not all nodes will be burned in the end.

Appendix B Identifying Configurations

For the sake of efficiency, all configurations are uniquely identified by a binary number. This increases the speed when inspecting whether a configuration has been found by the algorithm before. The system of identification works by labeling and ordering each edge on the cut graph. In that same order a number is given to each edge, 1 if this edge appears on the configuration and 0 otherwise (Fig. B1). The sequence of 1's and 0's will thus uniquely identify the configuration.

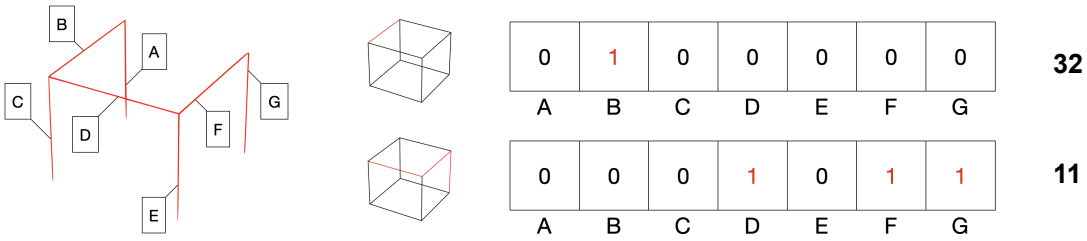

Fig. B1 Identification of configurations. Configurations are uniquely identified by a binary number. The edges on the shell cut graph (left) are labeled and ordered, in this case with letters from A to G. Binary numbers for each configuration are constructed by pairing 1's and 0's with the letters that identify the edge, 1 if that edge is on the configuration and 0 otherwise. The top configuration, is a one edge configuration where only edge B is present, thus the binary number 0100000 (32 in decimal) uniquely identifies this structure. The bottom configuration is composed of edges D, F and G, following the same process the number 0001011 is constructed (11 in decimal).

Appendix C Misfolds

The input to the algorithm does not need to be a template and a folded polyhedron. Any structure obtained through the process of folding can be used as input to the algorithm, for example a misfold. Misfolds are defined as structures in which a bond needs to be removed in order to correctly fold it.

On Fig. 5.B we show the folding landscape of the cube for every possible template as we did on Fig. 4 (in blue), this time we added misfolded states as target structures (in orange) and their intermediate folding states (in yellow) as well. These misfolded states were obtained by careful observation of a macro-meter scale kirigami when subject to different folding and binding events. These misfolded states with zero red links were used as the final configuration inputs to the algorithm as well as their respective initial configurations (templates). After this, we search for isomorphic structures among the whole catalog of found folding states and construct the map of folding exactly as we did before. In this way, we are able to see the whole evolution of the folding process until it reaches a misfolded state, these states (yellow) are states which can no longer evolve through folding alone to the correctly folded structure which is the cube.

On Fig. 5.A we present the folding map of the multifarious configurations (in blue), we have also added three misfolded configurations as inputs (in orange). Similarly to the previous figure we searched for isomorphic folding states and grouped them into one, the resulting map gives information on the key cuts and key binds after which the structure can no longer evolve towards the correctly folded structure.

Appendix D Efficiency

The algorithm was also applied to the Icosahedron and the Dodecahedron, two of the most complex Platonic solids which have 43380 templates each. To find all possible configurations for the unfolding of one of these polyhedra is a computationally costly process but a possible one. Along with the thousands of configurations found, the rate of discovery for new configurations was also studied regarding the number of templates given to the algorithm. The idea was to construct the folding map of several

templates and then combine them one by one, searching and eliminating equivalent configurations as explained before. On Fig. D1 we can see that the number of newly found configurations scales sublinearly with the number of added templates. This is an indication that the algorithm is efficient at probing the configurational space and that many common configurations exist in the folding process of different templates. We can also observe that the rate of newly found configurations starts to decrease at a certain point close to a thousand templates. This is an indication that something similar to a plateau is to be reached, and the configurational space has already been mostly explored, with new configurations making fewer and fewer contributions to the configurational space.

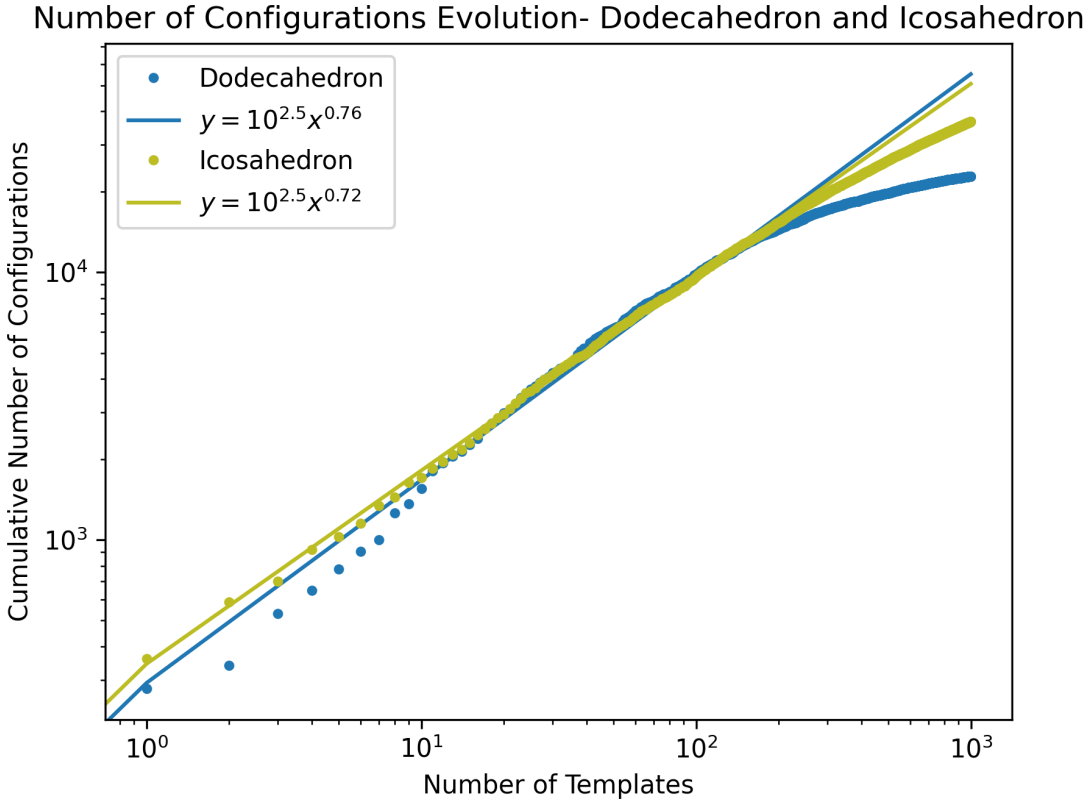

Fig. D1 Dependency of the cumulative number of configurations with the number of templates. Scaling of the newly found intermediate configurations with the number of analyzed templates of the two most complex platonic solids, the icosahedron and the dodecahedron, both with more than 48 thousand templates each. The number of newly found configurations scales, at first sight, sublinearly with the number of analyzed templates. In the end a deceleration is observed, indicative of a stagnation in the rate of newly found configurations. The configurational space seems to already be well-probed after a certain magnitude of templates are analyzed.

Another measure of efficiency is the number of iterations the algorithm takes to find the complete folding landscape for a given template and its target structure. If one was to verify all possible intermediate configurations made only by combining different edges to be cut, the pool of configurations to verify would scale factorially with the number of cuts needed to open the structure. As the number of cuts is proportional to the size of the system, we plot on Fig. D2 the factorial scaling and the number of iterations of the algorithm for different templates of polyhedra with variable size. The number of iterations needed to probe the whole folding landscape scales slower than the factorial scaling. This means that the pool of structures to verify is much less with our algorithm than it would be with simple combinatorial methods.

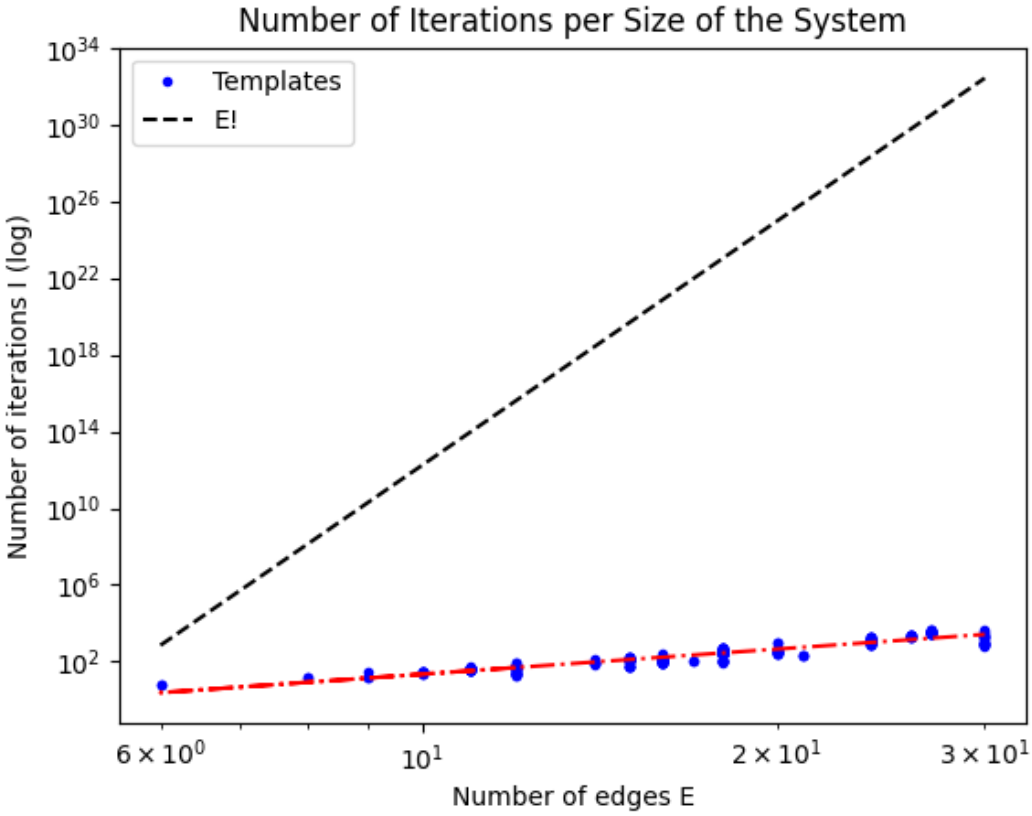

Fig. D2 Dependency of the number of iterations of the algorithm with the size of the system. Comparison between the scaling of the number of iteration of the algorithm for different templates of variable size and the factorial explosion of most combinatorial methods. Templates of different polyhedra were used as inputs to the algorithm and the number of iterations of the algorithm was computed against the size of the polyhedra, ie the number of edges. Our algorithm significantly decreases the number of iterations needed to probe the complete folding landscape comparing to simple combinatorial exploration.

Appendix E Dodecahedron

On Fig. E1 the folding network for one template of the dodecahedron is shown.

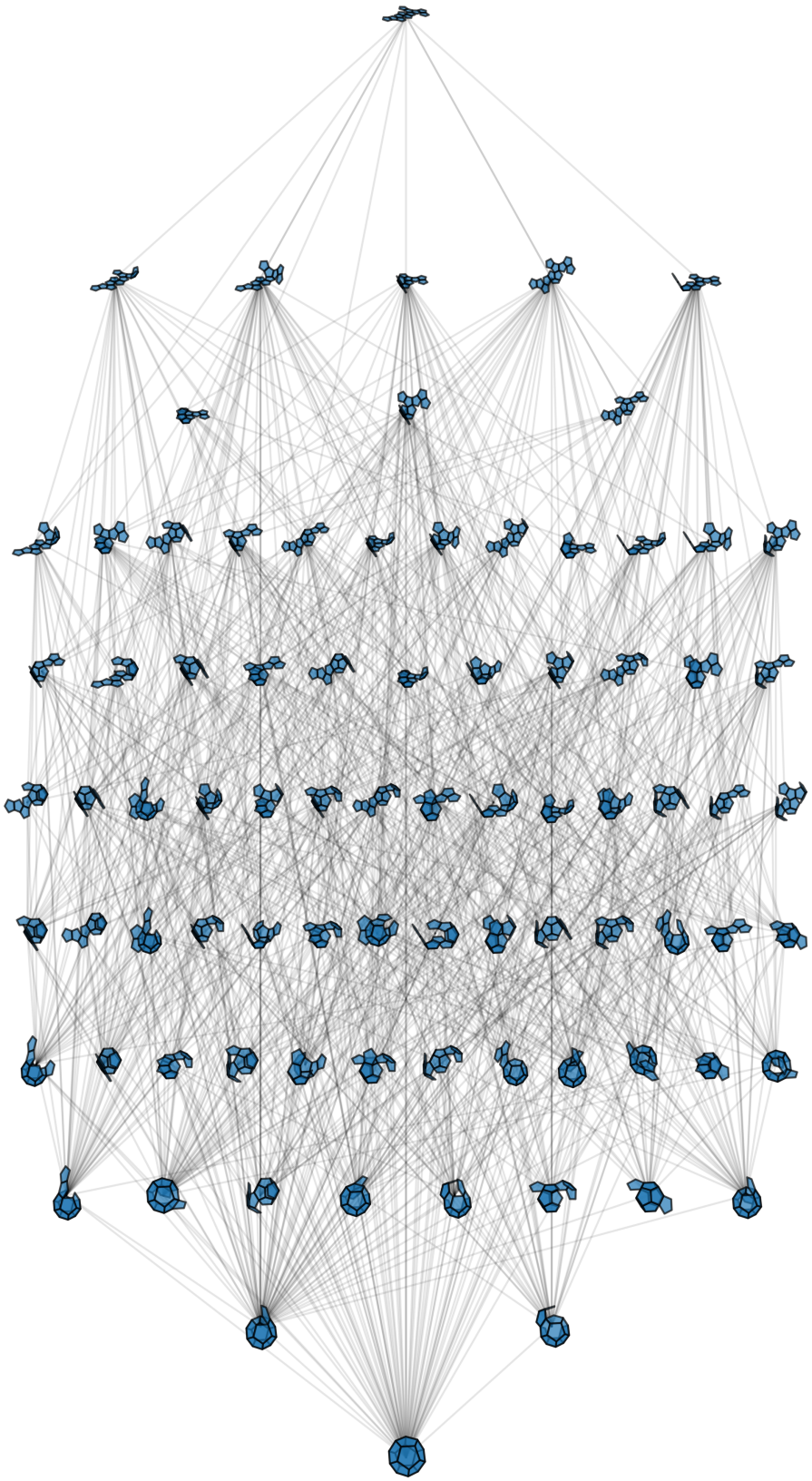

Fig. E1 Folding landscape for one template of the dodecahedron. As in Figures 4 and 5, the network of intermediate folding states is here shown for a specific template of the dodecahedron, its target structure. The resulting folding network is much more complex than the previously shown due to the increase on the number of edges of the target structure and consequently on the cut graph. Despite the complexity of the polyhedron we are still able to probe its folding landscape with success.

References

[1] Qi, J. *et al.* Recent progress in active mechanical metamaterials and construction principles. *Advanced Science* **9**, 2102662 (2022). URL https://onlinelibrary.wiley.com/doi/abs/10.1002/advs.202102662.

[2] Choi, G. P. T. Computational design of art-inspired metamaterials. *Nature Computational Science* **4**, 549–552 (2024). URL https://doi.org/10.1038/s43588-024-00671-y.

[3] Chen, S., Chen, J., Zhang, X., Li, Z.-Y. & Li, J. Kirigami/origami: unfolding the new regime of advanced 3d microfabrication/nanofabrication with "folding". *Light: Science & Applications* **9**, 75 (2020). URL https://doi.org/10.1038/s41377-020-0309-9.

[4] Felton, S., Tolley, M., Demaine, E., Rus, D. & Wood, R. A method for building self-folding machines. *Science* **345**, 644–646 (2014). URL https://www.science.org/doi/abs/10.1126/science.1252610.

[5] Gu, H. *et al.* Self-folding soft-robotic chains with reconfigurable shapes and functionalities. *Nature Communications* **14**, 1263 (2023). URL https://doi.org/10.1038/s41467-023-36819-z.

[6] Pedivellano, A. & Pellegrino, S. Folding kinematics of kirigami-inspired space structures. *International Journal of Solids and Structures* **300**, 112865 (2024). URL https://www.sciencedirect.com/science/article/pii/S0020768324002245.

[7] Zirbel, S. A. *et al.* Accommodating thickness in origami-based deployable arrays1. *Journal of Mechanical Design* **135**, 111005 (2013). URL https://doi.org/10.1115/1.4025372.

[8] Mitani, J. Pillow box design (2024). Preprint at https://arxiv.org/abs/2410.17593, 2410.17593.

[9] Zhu, Y. & Filipov, E. T. Large-scale modular and uniformly thick origami-inspired adaptable and load-carrying structures. *Nature Communications* **15**, 2353 (2024). URL https://doi.org/10.1038/s41467-024-46667-0.

[10] Babaee, S. *et al.* Kirigami-inspired stents for sustained local delivery of therapeutics. *Nature Materials* **20**, 1085–1092 (2021). URL https://doi.org/10.1038/s41563-021-01031-1.

[11] Huang, H.-W. *et al.* Adaptive locomotion of artificial microswimmers. *Science Advances* **5**, eaau1532 (2019). URL https://www.science.org/doi/abs/10.1126/sciadv.aau1532.

[12] McMullen, A., Muñoz Basagoiti, M., Zeravcic, Z. & Brujic, J. Self-assembly of emulsion droplets through programmable folding. *Nature* **610**, 502–506 (2022). URL https://doi.org/10.1038/s41586-022-05198-8.

[13] Veneziano, R. *et al.* Role of nanoscale antigen organization on b-cell activation probed using dna origami. *Nature Nanotechnology* **15**, 716–723 (2020). URL https://doi.org/10.1038/s41565-020-0719-0.

[14] Kim, M. *et al.* Harnessing a paper-folding mechanism for reconfigurable dna origami. *Nature* **619**, 78–86 (2023). URL https://doi.org/10.1038/s41586-023-06181-7.

[15] Kuribayashi-Shigetomi, K., Onoe, H. & Takeuchi, S. Cell origami: Self-folding of three-dimensional cell-laden microstructures driven by cell traction force. *PLOS ONE* **7**, 1–8 (2012). URL https://doi.org/10.1371/journal.pone.0051085.

[16] Lamoureux, A., Lee, K., Shlian, M., Forrest, S. R. & Shtein, M. Dynamic kirigami structures for integrated solar tracking. *Nature Communications* **6**, 8092 (2015). URL https://doi.org/10.1038/ncomms9092.

[17] Yasuda, H. & Yang, J. Reentrant origami-based metamaterials with negative poisson's ratio and bistability. *Phys. Rev. Lett.* **114**, 185502 (2015). URL https://link.aps.org/doi/10.1103/PhysRevLett.114.185502.

[18] Wang, Z. *et al.* Origami-based reconfigurable metamaterials for tunable chirality. *Advanced Materials* **29**, 1700412 (2017). URL https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201700412.

[19] Fang, H., Chu, S.-C. A., Xia, Y. & Wang, K.-W. Programmable self-locking origami mechanical metamaterials. *Advanced Materials* **30**, 1706311 (2018). URL https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201706311.

[20] Tao, J., Khosravi, H., Deshpande, V. & Li, S. Engineering by cuts: How kirigami principle enables unique mechanical properties and functionalities. *Advanced Science* **10**, 2204733 (2023). URL https://onlinelibrary.wiley.com/doi/abs/10.1002/advs.202204733.

[21] Li, Y., Zhang, Q., Hong, Y. & Yin, J. 3d transformable modular kirigami based programmable metamaterials. *Advanced Functional Materials* **31**, 2105641 (2021). URL https://onlinelibrary.wiley.com/doi/abs/10.1002/adfm.202105641.

[22] Zheng, X. *et al.* Ultralight, ultrastiff mechanical metamaterials. *Science* **344**, 1373–1377 (2014). URL https://www.science.org/doi/abs/10.1126/science.1252291.

[23] Guo, X. *et al.* Designing mechanical metamaterials with kirigami-inspired, hierarchical constructions for giant positive and negative thermal expansion. *Advanced Materials* **33**, 2004919 (2021). URL https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/adma.202004919.

[24] Li, Z., Chen, W., Hao, H., Yang, Q. & Fang, R. Energy absorption of kirigami modified corrugated structure. *Thin-Walled Structures* **154**, 106829 (2020). URL https://www.sciencedirect.com/science/article/pii/S0263823120307072.

[25] Zhang, Q. *et al.* Impact behavior of corrugated-core infilling foam sandwich composite structure. *Case Studies in Construction Materials* **17**, e01418 (2022). URL https://www.sciencedirect.com/science/article/pii/S2214509522005502.

[26] Fathers, R., Gattas, J. & You, Z. Quasi-static crushing of eggbox, cube, and modified cube foldcore sandwich structures. *International Journal of Mechanical Sciences* **101-102**, 421–428 (2015). URL https://www.sciencedirect.com/science/article/pii/S0020740315003021.

[27] Sussman, D. M. *et al.* Algorithmic lattice kirigami: A route to pluripotent materials. *Proceedings of the National Academy of Sciences* **112**, 7449–7453 (2015). URL https://www.pnas.org/doi/abs/10.1073/pnas.1506048112.

[28] An, N., Domel, A. G., Zhou, J., Rafsanjani, A. & Bertoldi, K. Programmable hierarchical kirigami. *Advanced Functional Materials* **30**, 1906711 (2020). URL https://onlinelibrary.wiley.com/doi/abs/10.1002/adfm.201906711.

[29] Neville, R. M., Scarpa, F. & Pirrera, A. Shape morphing kirigami mechanical metamaterials. *Scientific Reports* **6**, 31067 (2016). URL https://doi.org/10.1038/srep31067.

[30] O'Rourke, J. *How to Fold It: The Mathematics of Linkages, Origami, and Polyhedra* (Cambridge University Press, 2011).

[31] Demaine, E. D. & O'Rourke, J. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra* (Cambridge University Press, 2007).

[32] Dodd, P. M., Damasceno, P. F. & Glotzer, S. C. Universal folding pathways of polyhedron nets. *Proceedings of the National Academy of Sciences* **115**, E6690–E6696 (2018). URL https:

//www.pnas.org/doi/abs/10.1073/pnas.1722681115.

[33] Kaplan, R., Klobušický, J., Pandey, S., Gracias, D. H. & Menon, G. Building Polyhedra by Self-Assembly: Theory and Experiment. *Artificial Life* **20**, 409–439 (2014). URL https://doi.org/10.1162/ARTL_a_00144.

[34] Johnson-Chyzhykov, D. & Menon, G. The Building Game: From Enumerative Combinatorics to Conformational Diffusion. *Journal of Nonlinear Science* **26**, 815–845 (2016). URL https://doi.org/10.1007/s00332-016-9291-z.

[35] Araújo, N. A. M., da Costa, R. A., Dorogovtsev, S. N. & Mendes, J. F. F. Finding the optimal nets for self-folding kirigami. *Phys. Rev. Lett.* **120**, 188001 (2018). URL https://link.aps.org/doi/10.1103/PhysRevLett.120.188001.

[36] Coniglio, A. Thermal phase transition of the dilute s-state potts and n-vector models at the percolation threshold. *Phys. Rev. Lett.* **46**, 250–253 (1981). URL https://link.aps.org/doi/10.1103/PhysRevLett.46.250.

[37] Herrmann, H. J., Hong, D. C. & Stanley, H. E. Backbone and elastic backbone of percolation clusters obtained by the new method of 'burning'. *Journal of Physics A: Mathematical and General* **17**, L261 (1984). URL https://dx.doi.org/10.1088/0305-4470/17/5/008.

[38] Jüttner, A. & Madarasi, P. Vf2++—an improved subgraph isomorphism algorithm. *Discrete Applied Mathematics* **242**, 69–81 (2018). URL https://www.sciencedirect.com/science/article/pii/S0166218X18300829.

[39] Hagberg, A. A., Schult, D. A. & Swart, P. J. Varoquaux, G., Vaught, T. & Millman, J. (eds) *Exploring network structure, dynamics, and function using networkx.* (eds Varoquaux, G., Vaught, T. & Millman, J.) *Proceedings of the 7th Python in Science Conference*, 11 – 15 (Pasadena, CA USA, 2008).

[40] Pandey, S. *et al.* Algorithmic design of self-folding polyhedra. *Proc. Natl. Acad. Sci. U.S.A.* **108**, 19885–19890 (2011). URL https://doi.org/10.1073/pnas.1110857108.

[41] Melo, H. P. M., Dias, C. S. & Araújo, N. A. M. Optimal Number of Faces for Fast Self-Folding Kirigami. *Commun. Phys.* **3**, 154 (2020). URL https://doi.org/10.1038/s42005-020-00423-0.

[42] Murugan, A., Zeravcic, Z., Brenner, M. P. & Leibler, S. Multifarious assembly mixtures: Systems allowing retrieval of diverse stored structures. *Proceedings of the National Academy of Sciences* **112**, 54–59 (2015). URL https://doi.org/10.1073/pnas.1413941112.

[43] Pinto, D. E. P., Araújo, N. A. M., Šulc, P. & Russo, J. Inverse design of self-folding 3d shells. *Phys. Rev. Lett.* **132**, 118201 (2024). URL https://doi.org/10.1103/PhysRevLett.132.118201.