Automating Rigid Origami Design

Jeremia Geiger , **Karolis Martinkus** , **Oliver Richter** and **Roger Wattenhofer**
ETH Zurich
jeremia.s.r.geiger@gmail.com, {martinkus, richtero, wattenhofer}@ethz.ch

Abstract

Rigid origami has shown potential in large diversity of practical applications. However, current rigid origami crease pattern design mostly relies on known tessellations. This strongly limits the diversity and novelty of patterns that can be created. In this work, we build upon the recently developed principle of three units method to formulate rigid origami design as a discrete optimization problem, the rigid origami game. Our implementation allows for a simple definition of diverse objectives and thereby expands the potential of rigid origami further to optimized, application-specific crease patterns. We showcase the flexibility of our formulation through use of a diverse set of search methods in several illustrative case studies. We are not only able to construct various patterns that approximate given target shapes, but to also specify abstract, function-based rewards which result in novel, foldable and functional designs for everyday objects.

1 Introduction

Origami may be ancient to the arts, but it is young to science. Only in recent years, researchers began to rigorously investigate the underlying principles of folding. This has resulted in applications of origami in various fields of robotics, architecture, biomedical engineering, deployable structures, metamaterials, aerospace applications, and more [Meloni *et al.*, 2021; Wang, 2019; Callens and Zadpoor, 2018; Turner *et al.*, 2016; Kuribayashi *et al.*, 2006; Morgan *et al.*, 2016].

Rigidly foldable patterns, known as rigid origami, are of particular interest in practical applications since the material does not deform while folding such patterns. Even more importantly, in principle the folding motion of a rigid pattern can be induced by a single folding activation, also referred to as the patterns' degree of freedom (DOF) [He and Guest, 2018; Zimmermann *et al.*, 2020]. This allows for creation of shapes that are not only beautiful, but are also functional.

Traditional origami tackles the question of how a given crease pattern, defined by a set of creases drawn on a flat piece of paper, will fold. However, a more useful question is how to define a pattern such that it folds into a given target shape. This is commonly known as the inverse origami problem. Note that this is a complex problem since determining the foldability of a general crease pattern alone is NP-hard [Demaine *et al.*, 2016]. Recent works started to address the inverse origami problem [Demaine and Tachi, 2017; Dudte *et al.*, 2016; He and Guest, 2018]. Unfortunately, these previous works either focus on non-rigid origami and allow the pattern to have many degrees of freedom [Demaine and Tachi, 2017] or approximate shapes using only known tessellations and their variations [Dudte *et al.*, 2016; He and Guest, 2018; Tachi, 2010]. Even though it is possible to approximate some shapes very well using rigid origami tessellations, beyond the space of the tessellations there is a much larger space of general foldable patterns. Furthermore, none of these works explore the possibility of specifying not the target shape itself, but a flexible abstract objective or function that the folded shape should fulfill, which allows for even more creative freedom.

In this work, we formulate rigid origami design as a discrete optimization problem , which allows us to search for general rigidly foldable patterns which approximate a given shape or minimize some other user-specified objective function [McAdams and Li, 2014] using a diverse set of search methods. We treat the crease pattern as a graph where edges represent the crease lines and vertices their intersections [He and Guest, 2019]. We place vertices one after the other on a pre-defined grid to construct these graphs, see Figure 1. Our formulation brings the optimization problem close to combinatorial problems, such as board games, where Reinforcement learning (RL) and other techniques have demonstrated great success [Silver *et al.*, 2016; Silver *et al.*, 2018; Vinyals *et al.*, 2019]. We first derive the methodology characterizing the game environment and then expose it to various types of artificial agents for validation. Our results highlight the flexibility of our formulation and open up the possibility for future research – both in rigid origami design and optimization methods for our application.

2 Related Work

In its most primitive form, a crease pattern consists of a single, center vertex and multiple leaf vertices connected to the center vertex. The fundamental origami theorems of Maekawa and Kasahara [1983] and Kawasaki [1991] deter-

Code is available at https://github.com/belalugaX/rigid-origami

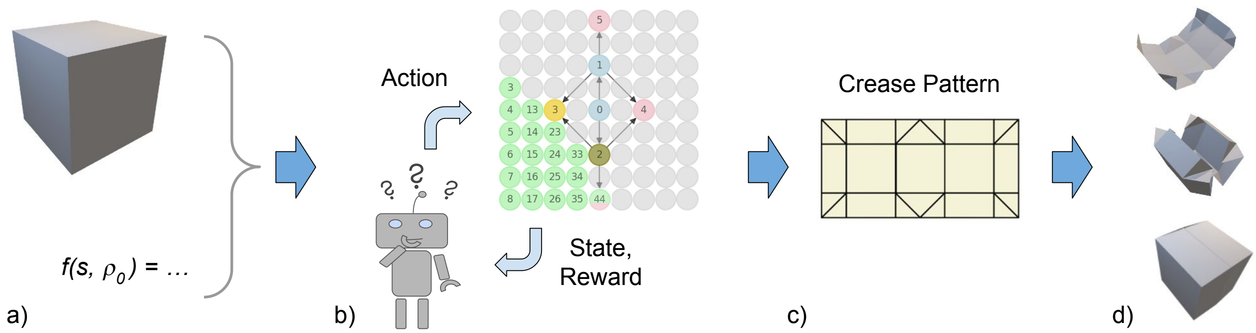

Figure 1: Overview of our approach. Given a target shape or more generally a target objective function such as desired volume (a), we formulate the inverse origami problem as a token place game on a checkerboard (b). An agent can iteratively interact with this game formulation and improve its policy with respect to the objective. The best rollout can be converted into a crease pattern (c), which can be used to rigidly fold a flat sheet into the corresponding shape that maximizes the objective (d).

mine flat foldability of such a primitive pattern, based on two characteristics of the edges: their direction of folding (the mountain-valley assignment), and their spanned planar angles (the sector angles). The foldability of multi-vertex patterns, however, has been proven to be an NP-hard problem [Demaine *et al.*, 2016], which means that there exists no efficient algorithm guaranteed to find a solution. Particularly interesting are non-developable surfaces (curved shapes), which are those that cannot be unfolded into a plane sheet.

Nevertheless, with the Origamizer, Demaine and Tachi [2017] propose the state-of-the-art algorithm for solving the inverse origami problem for arbitrary target shapes. In particular, solving the inverse problem means being able to generate a crease pattern, which when folded, approximates the surface of the target up to a fixed precision. As pointed out by He and Guest [2018], the Origamizer has remarkable abilities to approximate complex surfaces, but also limitations regarding our scope of rigid origami: the generated patterns have many DOF, require a sequential folding process and favor tugged panels when approximating non-developable surfaces. This makes the Origamizer solutions impractical for many engineering problems.

It has not yet been fully understood how to design rigidly foldable patterns with limited DOF [He and Guest, 2018; Turner *et al.*, 2016]. Thus, other state-of-the-art methods approach shape approximation from a different paradigm: It is a common practice in origami to modify a few well-known tessellation patterns, such as Miura Ori, Waterbomb-, Yoshimura-, and Resch's pattern [Miura, 1969; Hanna *et al.*, 2014; Yoshimura, 1955; Resch, 1970]. Their properties are summarized and described by Tachi [2010]. In comparison to the Origamizer, methods based on known tessellations [Tachi, 2010; He and Guest, 2018; Callens and Zadpoor, 2018; Dudte *et al.*, 2016] have an inherent rigid foldability and limited DOF. However, the main limitation of these methods is that they rely on the modified tessellations, strongly limiting the potential design space of patterns. This leads to some authors restricting themselves to approximating particular kinds of surfaces such as cylindrical ones [Dudte *et al.*, 2016]. As pointed out by [Turner *et al.*, 2016; Meloni *et al.*, 2021], the future prospect of origami for engineering applications relies on methods for the design of new and original rigid foldable patterns with properties specific to their field of application.

A general design method for rigidly foldable patterns has not yet been proposed [Meloni *et al.*, 2021; Turner *et al.*, 2016]. Nevertheless, the recently proposed principle of three units (PTU) method [Zimmermann *et al.*, 2020; Zimmermann and Stanković, 2020] allows for the rule-based generation and efficient simulation of rigidly foldable patterns. The PTU implies a set of rules, such that a pattern complying with those rules is guaranteed to fold rigidly. Although the PTU does have limitations, in particular a restriction to acyclic origami graphs, it still enables the rule-based generation and efficient folding simulation of rigidly foldable patterns in large configuration spaces.

We seek to formulate the graph design problem in a reinforcement learning (RL) setting to leverage the spectacular success of RL methods on various problems with large search spaces [Silver *et al.*, 2016; Silver *et al.*, 2018; Vinyals *et al.*, 2019]. RL has also been applied to various design tasks, such as designing molecules [Simm *et al.*, 2020a; Simm *et al.*, 2020b] or designing computer chips [Mirhoseini *et al.*, 2021]. However, to the bests of our knowledge, there have been no attempts so far to apply RL to rigid origami pattern discovery and the inverse origami problem in general.

Our approach also allows for a more interesting direction than simply approximating target shapes: discovering origami patterns with desired properties, such as a given surface area and height of the folded pattern. In a very nascent example of this, a specifically formulated genetic algorithm has been used to discover a simple pattern that folds a square sheet into a smaller flat sheet that has a predetermined area using a limited number of crease lines [McAdams and Li, 2014]. We explore this in a much more general form. Where our purposed method can tackle a wide array of objectives and thus applications.

3 Rigid Origami as Discrete Optimization

We formulate the task of designing a rigid origami crease pattern as a game in which an agent iteratively constructs the

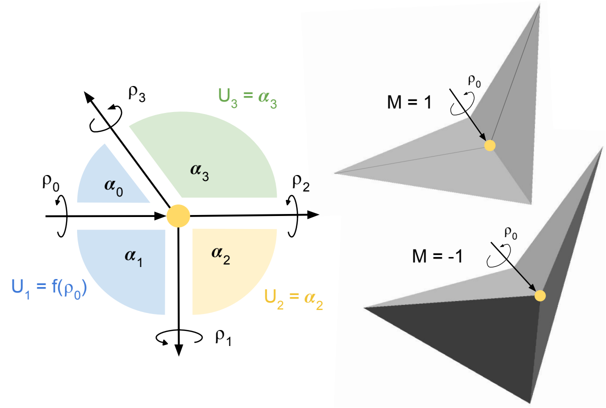

Figure 2: A degree 4 vertex with one incoming and 3 outgoing edges. Indicated are the sector angles α, the folding (dihedral) angles ρ and the unit angles U_1, U_2 and U_3. The illustrations on the right show the effect of the rigid body mode $M \in \{-1, 1\}$ for the same input angle ρ_0.

crease pattern by taking actions and receiving observations and rewards in response. Such an abstract formulation admits not only the use of reinforcement learning and evolutionary approaches but also constraint satisfaction search methods that explore the game tree.

3.1 Notation

We denote a crease pattern's state at time step t by $s_t \in \mathcal{S}$, where $\mathcal{S}$ denotes the space of developable crease patterns which can be unfolded into a plane sheet. Every state $s \in \mathcal{S}$ can further be represented by the corresponding crease pattern graph $s = G(V, E)$, where V is the set of vertices where crease lines meet and E is the set of directed edges along the crease lines.

Every degree-δ vertex $v_i \in V$ is characterized by a set of sector angles $\boldsymbol{\alpha} = \{\alpha_i\}$ and a rigid body mode M, while every crease line $e_i \in E$ has a folding (dihedral) angle $\{\rho_i\}$ assigned to it. Here $\alpha_i \in (0, 2\pi)$, $\rho_i \in (-\pi, \pi)$, and $M \in \{-1, 1\}$. Figure 2 shows an illustration of these definitions for a simple degree 4 vertex. Sector angles $\boldsymbol{\alpha}$ remain fixed during the folding motion (as panels are rigid), whereas the dihedral angles $\boldsymbol{\rho}$ change as the object folds according to the dynamics implied by the crease pattern. Throughout the paper, we limit our patterns to fold within a single degree of freedom (DOF), represented by the driving angle ρ_0. Note that patterns with a single driving angle are particularly appealing, as we only have to actuate one hinge to fold the whole object. The solution of vertex kinematics for any given configuration depends on the mountain or valley assignment of the creases. This property is expressed by the rigid body mode M.

3.2 PTU Kinematic Model

The PTU [Zimmermann *et al.*, 2020] provides both general rules for expanding an embedded crease pattern s, and an efficient kinematic model for folding simulation.

We do allow for multiple sources in some of our starting configurations, however, all sources have the same driving angle ρ_0.

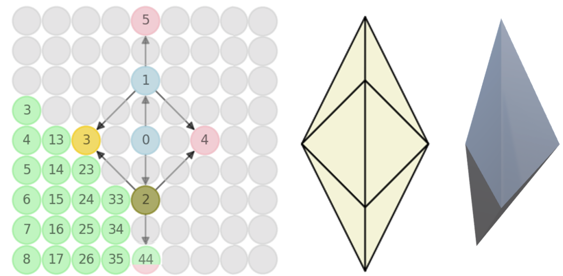

Figure 3: Origami game in a particular state (left: board, middle: creased paper, right: folded shape). The vertices on the board are color-coded as follows: Blue/Olive vertices have been extended, with the color indicating the rigid body mode M, pink vertices are extendable and the yellow vertex is currently selected for an extension. Green slots indicate permitted and the grey slots forbidden locations for the first outgoing edge endpoint – these indicate the available actions. Note that node 44 is colored pink and green, as it is an extendable vertex that can be selected as an endpoint.

Specifically, for a directed acyclic graph $G(V, E)$ the PTU kinematic model allows us to sequentially solve for all folding angles $\boldsymbol{\rho}$, starting from the graph origin and finishing at the leaves.

In its essence the PTU is based on the following observation: for a given vertex, its rigid folding motion is kinematically determinate if all but three of the adjacent edges' driving angles ρ_i are known (incoming edges). The angles spanned between the three outgoing edges then define the three unit angles U_1, U_2, and U_3, which span a spherical triangle on the unit sphere. The PTU kinematic model now provides the formalism to solve for the outgoing edges' driving angles ρ_j by means of spherical trigonometry. Hence the vertex' folding motion is fully defined.

Therefore, if we are given the driving angle ρ_0 and a directed acyclic graph as a crease pattern, we can sequentially calculate all unknown folding angles. As noted by Zimmermann *et al.* (2020), this only involves forward kinematics and is thereby computationally cheaper than other methods that have to rely on matrix inversions [Tachi, 2010]. We refer an interested reader to [Zimmermann *et al.*, 2020] for the details of the kinematic model and focus on the graph construction here.

3.3 The Origami Game

Given the constraints of the PTU method, we aim to formulate the task of finding a suitable crease pattern s as a discrete optimization problem. More specifically, we sequentially construct a directed acyclic graph $G(V, E)$ by placing vertices v and connecting crease lines e on a grid board of fixed size. The board represents the unfolded paper and the discrete nature of the grid allows for efficient exploration of the infinite space of possible crease patterns.

In this formulation, we can also view the task of designing a crease pattern as playing a single-player game in which the player sequentially places vertices on the board. The board is first initialized with a simple pre-selected starting pattern, such as a line or a square, for which the size and the position can be chosen by the agent. Then, in the first phase of each turn, the player chooses a vertex that has only incoming edges so far, i.e. an extendable vertex. In the second phase, the player then chooses three locations on the board, which define the three outgoing edges of the vertex chosen before. A new vertex is created in every chosen location that did not have a vertex so far and the player is asked again to choose the next vertex to extend. In the first phase of each turn, the player can also be given the option to choose a special *source* action instead of selecting an extendable vertex. This action places a new source vertex in an empty space on the board, which receives the same external driving angle ρ_0 as the original source vertex, thus keeping the 1-DOF constraint. This can make for easier construction of certain shapes.

The player reaches the end of an episode in the game if either (1) a special *terminate* action is played, (2) some non-foldable game state is encountered, or (3) there are no more actions available in the current game state. See Figure 3 for an illustration of the game.

3.4 Rules and Constraints

Apart from an efficient search space discretization, our board game allows us to enforce certain rules and constraints given by the PTU kinematic model and the rigidity of the panels being folded.

Specifically, the PTU provides a kinematic model for the folding motion of extended vertices of arbitrary degree δ, given the following constraints:

1. The vertex has exactly three outgoing edges and $\delta - 3$ incoming edges.
2. The triangle inequality for spherical triangles must be fulfilled throughout the entire folding motion: $U_{min} + U_{med} \geq U_{max}$, where $U_{min}, U_{med},$ and U_{max} represent the smallest, middle and largest unit angle respectively.
3. The crease pattern $G(V, E)$ must be acyclic.

Further, as we aim to fold rigid panels, we can infer that:

4. The crease pattern graph $G(V, E)$ must be planar, as crossing edges would result in overlapping panels in the flat folded starting state.
5. Panels must not intersect with other panels during the entire folding motion.

All constraints 1-5 are either geometrical (1,2,4,5) or topological (3) in nature. In the discrete action space, we can impose constraints 1-4 a priori through action masking, such that an agent is not eligible to take any action that would lead to a violating state. The non-intersection of faces constraint 5, however, cannot be constrained prior to the computation of the kinematics. Hence this constraint can only be evaluated after the full extension of a vertex.

In detail, we impose the constraints 1-5 as follows:

1. By extending each vertex with exactly three child vertices.
2. By computing the triangle inequality for spherical triangles after the second and before the third extension of a vertex for every permitted action. We mask the actions accordingly.
3. By masking extended vertices from the available actions.
4. By performing an edge-edge intersection test for available actions and masking accordingly.
5. By performing Moeller's Fast Triangle-Triangle Intersection Test [Möller, 1997] for a set of folding angles that mirror the folding motion. If panels intersect, we roll back to the last foldable state (the state before the failed extension) and terminate the episode.

3.5 Guiding the Search

Additionally to the methodological constraints outlined above which guarantee rigid foldability, our formulation also allows incorporating additional constraints to guide the search in various aspects. This allows the user to influence and tune the result more precisely or to explore shapes with different characteristics. Specifically:

1. The board size can be chosen based on the desired complexity of the resulting shape. Smaller board sizes yield simpler solutions while larger board sizes allow for more complex shapes.
2. Desired symmetries of the resulting shape can be incorporated by duplicating actions across symmetry axes.
3. If a good starting pattern is known, the board can be seeded with the corresponding graph and extended from there. This can also be used to modify previously discovered or handcrafted patterns, by deleting parts of them and using the rest as a starting pattern.
4. Finally, we can easily mask actions based on a maximal permitted crease length.

3.6 Objectives

An important advantage of the formulation as an optimization problem is the freedom to choose the objective function. Instead of defining how the pattern should be folded, we can now focus on what we want from the resulting pattern and leave the implementation to the optimization. Specifically, given the general objective

$$\arg\max f(s, \rho_0)$$

we can design the objective function $f : \mathcal{S} \times (-\pi, \pi) \to \mathbb{R}$ to reflect our requirements.

Before we give examples of explicit objective functions, we first discuss the formulation in light of the sequential construction of a solution. That is, we seek to reward every partial solution at time step t with r_t, such that $f(s_T, \rho_0) = \sum_{t=0}^{T} r_t$. Here, T denotes the time step in which the episode terminates. For general objectives f we can always achieve this by setting $r_T = f(s_T, \rho_0)$ and $r_{t \neq T} = 0$. However, both RL and tree search methods benefit from early feedback. Specifically, if we wish to cut a branch in a game tree traversal at the partial solution $s_{t'}$, we can do so if

1. $r_t \leq 0\ \forall t$, that is, any partial solution can only get worse

Note that this also implies that $f(s, \rho_0) \leq 0$ for any state s and driving angle ρ_0. However, this can often be easily achieved by subtracting a constant that upper bounds the original objective function and using the result as an objective function.

2. $\sum_{t=0}^{t'} r_t < f(s^*, \rho_0)$, where s^* is the best solution so far.

Generally, we can shape rewards as $\tilde{r}_t = r_t + \Delta\Phi_t$ by adding a difference in potential $\Delta\Phi_t = \Phi(s_t, \rho_0) - \Phi(s_{t-1}, \rho_0)$ without changing the objective [Ng *et al.*, 1999]. However, to fulfill requirement (1) above we seek a potential function Φ that is monotonically decreasing as the game progresses, i.e. $\Phi(s_t, \rho_0) \leq \Phi(s_{t-1}, \rho_0)$. Luckily, many objectives in our setup can be naturally decomposed into a part that is monotonically decreasing with t and a remainder, which can be given at the end of the episode.

To validate our approach, we first focus on shape approximation, i.e. the inverse origami problem. We then provide a range of alternative objectives to highlight the flexibility of our approach.

Shape Approximation

The standard objective of the inverse origami problem is to find a crease pattern s such that the folded polyhedral manifold $\mathcal{M}_{origami} \subset \mathbb{R}^3$ approximates a given target shape $\mathcal{M}_{target} \subset \mathbb{R}^3$. As a measure for the accuracy of approximation, one commonly takes the Hausdorff distance $d(X, Y)$ between sets of points X and Y given by

$$d(X, Y) = \max\{d_\rightarrow(X, Y), d_\leftarrow(X, Y)\}$$

$$d_\rightarrow(X, Y) = \max_{x \in X}\left(\min_{y \in Y} ||x - y||_2\right)$$

$$d_\leftarrow(X, Y) = \max_{y \in Y}\left(\min_{x \in X} ||x - y||_2\right)$$

We fix a set of sampled target points $Y \sim \mathcal{M}_{target}$ and choose $X \subset \mathcal{M}_{origami}$ as detailed later. Note that the two terms $d_\rightarrow(X, Y)$ and $d_\leftarrow(X, Y)$ model two distinct aspects: The first measures how far the folded shape is from the target shape while the second measures how well the whole target shape is covered by the folded shape. As we only add points during the construction of a crease pattern, the furthest point from the target in the folded shape can only get worse. We can therefore set $\Phi(s_t, \rho_0) = -d_\rightarrow(P_t, Y)$, where $P_t \in \mathbb{R}^{3 \times |V_t|}$ denotes the spatial positions of the vertices V_t in s_t under driving angle ρ_0. We can use the vertex positions here as they reflect the corners and thereby the extremes of the folded polyhedran. The rewards are then given by $r_t = \Delta\Phi_t$ for $t \neq T$ and

$$r_T = -\max\{d_\rightarrow(P_T, Y), d_\leftarrow(X, Y)\} - \sum_{t=0}^{T-1} r_t$$

For r_T we sample X from the folded shape.

Abstract Rewards

If the reward function is more abstract and does not constrain the agent to matching a given shape as close as possible the agent can create (imagine) new shapes that accomplish the goal specified by the objective function. We sketch four kinds of handcrafted abstract objective functions to showcase the power of this approach. Specifically, starting from a flat sheet in the xy-plane at $z = 0$, we aim to fold everyday objects:

1. Bucket: To get a waterproof bucket, we maximize the smallest z-coordinate of the leaf nodes in the graph.
2. Shelf: Here we maximize the area of parallel planes in the folded shape, discarding solutions with less than three parallel planes.
3. Table: The table objective aims to get 4 points to a target z value (the legs) while keeping all other points in the $z = 0$ plane.
4. Chair: A chair that is symmetrical across the y-axis consists of a backrest (target z and y value) as well as at least three legs on the ground (also with target z and y values).

Note that these rewards can be augmented with certain desiderata such as a term that discourages points from sticking too far out. This augmentation can also be used for reward shaping with the same argumentation as before, points only get added, so the worst point can only get worse. More details and the exact formulation of all objectives are provided in Appendix F.

3.7 Driving Angle Optimization

The fitness of the final folded shape greatly depends on the driving angle ρ_0. Unfortunately, the optimal angle is usually not known before trying to fold the shape. In order to not constrain the agent to finding a pattern that is optimal given some particular driving angle, we instead specify a maximum driving angle $\rho_0^{\max}$ and keep track of 10 equally spaced driving angles from $0°$ to $\rho_0^{\max}$. During each reward calculation, we calculate the reward for each of the driving angles and give the highest of those rewards to the agent. If at any point a driving angle results in an intersection, we discard it. If there are no more driving angles that do not cause an intersection the episode terminates. Note that this optimization over the driving angle is optional since we can also provide ρ_0 if we know it upfront from domain knowledge. However, since we intend to highlight the general applicability of our approach in our experiments we include this optimization by default.

3.8 Search Methods

As with the objectives, our formulation as a discrete optimization problem also allows for a diverse range of search methods. Specifically, we compare the following approaches in the shape approximation task:

- Random search (RDM)
- Depth-first tree search (DFTS)
- Breadth-first tree search (BFTS)
- Monte Carlo tree search (MCTS)
- Proximal policy optimization (PPO)
- Evolutionary search (EVO)

RDM simply samples actions randomly from the available actions per step.

For DFTS and BFTS, the following applies: (1) actions are ordered randomly (2) we upper bound the branching factor per node to favor a broader search and (3) both search methods follow a branch and bound paradigm, such that visited

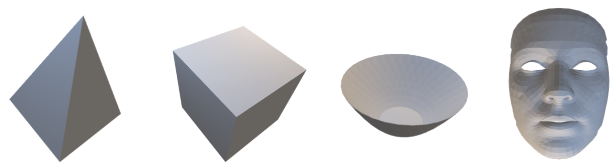

Figure 4: Visualization of the four target shapes used: cube, pyramid, bowl, and human face.

	Pyramid	Cube	Bowl	Human Face
RDM	$\mathbf{-0.09 \pm 0.00}$	-0.76 ± 0.13	-1.20 ± 0.16	-4.98 ± 0.35
DFTS	$\mathbf{-0.09 \pm 0.00}$	$\mathbf{-0.07 \pm 0.00}$	-0.85 ± 0.08	-4.38 ± 0.47
BFTS	$\mathbf{-0.09 \pm 0.00}$	-0.07 ± 0.01	$\mathbf{-0.75 \pm 0.05}$	-7.94 ± 0.16
MCTS	$\mathbf{-0.09 \pm 0.00}$	-0.79 ± 0.04	-1.09 ± 0.09	-5.01 ± 0.32
PPO	-0.09 ± 0.01	-0.68 ± 0.28	-1.24 ± 0.30	-5.70 ± 0.61
EVO	-0.17 ± 0.25	-0.44 ± 0.30	-1.09 ± 0.25	$\mathbf{-3.49 \pm 0.61}$

Table 1: Summary of best episode returns $f(\rho_0, s^*)$ across shapes and search methods. The highest returns are highlighted in bold.

nodes are pruned if their values are less than the best episode return seen throughout the search so far. DFTS searches upwards from the leaves of the game tree. BFTS samples a fixed number of available actions per node and then traverses along the child node with the highest value.

MCTS [Chaslot *et al.*, 2008] explores the game tree through random rollouts which get biased towards regions of higher returns. In accordance with best practices, we normalize the returns to $[-1, 1]$.

PPO [Schulman *et al.*, 2017] is a model-free deep reinforcement learning algorithm, which has seen a broad adaptation in the literature as a strong baseline. For our experiments, we rely on the PPO implementation by Liang *et al.* [2018].

For the evolutionary algorithm (EVO) we model agents as a simple list of action values, where the list length is proportional to the board size. The first half of the list is used in the node selection phase and the other is used in the extension phase. Actions are chosen greedily with respect to the listed values of the available actions. The best-performing agents are allowed to reproduce, such that the next generation consists of randomly perturbed copies of these agents as well as some new randomly initialized agents (newcomers).

See Appendix A for more details on all search methods and the hyperparameters used.

4 Experiments

First to to validate our general formulation and to benchmark different search methods we test the shape approximation task, on shapes of increasing complexity. See Figure 4 for a visualization of these targets and Appendix B for specific details. The first two shapes of cube and pyramid are developable (if cut open along the edges) and hence we expect that there exists an optimal pattern that can be modeled within the scope of our discrete environment and the PTU-based model. The bowl represents a complex, non-developable surface, that should result in more creative solutions. Finally we push the limitations of our formulation as we try to approximate the shape of a human face, which has very complex topography.

Secondly, we test our approach in free-form shape imagination. In this case only an abstract reward function is specified (Appendix F). We aim to capture the functionality of a few standard objects, such as a bucket (maximizing volume), table (maximizing surface area at a certain height) and a shelf (maximizing collective surface area over multiple heights).

All of the experiments were performed on a server with two 10-core Intel Xeon E5-2690 v2 CPUs and 125 GB of RAM. The environment is implemented in Open AI Gym [Brockman *et al.*, 2016].

4.1 Shape Approximation

We ran each search method for 500 thousand interactions with the environment, always keeping track of the best pattern found so far. To compare the search methods we evaluate the return $f(s^*, \rho_0)$ of the best pattern s^* that has been found. Since all our search methods are randomized algorithms, we run all searches for ten different random seeds and report the mean and the standard deviation. The results can be seen in Table 1. Qualitatively, we visualize the folded shapes of the best patterns found in any of the runs in Figure 5. See Appendix E for a visualization of the crease patterns and their folding motion. In the supplementary material we also provide videos of the folding motions.

Figure 5 shows that all methods are able to find the best possible approximation of the pyramid, where the slight opening at the top is an artifact of the fold angle discretization. However, for the cube, some methods struggle to find the optimal pattern. This highlights the difficulty of the optimization problem, as many crease patterns lead to well-separated local optima. Moving to the bowl we see that since the perfect approximation is now impossible, methods come up with more creative solutions. We note that DFTS and BFTS perform comparably well here, as they can leverage local optima to prune many unpromising paths. Finally, the results from the human face approximations show that an evolutionary algorithm performs the best if the search space becomes excessively large. We believe this stems from the principled exploration of multiple promising solutions in the evolutionary algorithm. In Appendix D we report the total number of environment interactions taken by each method to reach the best pattern.

In summary, we note that if the search space is moderately large, tree based search algorithms (i.e., DFTS and BFTS) perform well. However, these methods can become excessively expensive when the search space becomes larger and their initial exploration fails to make fast progress. In particular, BFTS struggles if the non-monotonic remainder of the objective is large relative to the monotonic part which is used to cut branches. Local optima are plentiful and well separated in the search space, leading policy-based methods (MCTS and PPO) to converge to a suboptimal solution prematurely. Note that the policy-based solutions generally arrive faster at their best solution, but in the end achieve worse accuracy. In contrast, the evolutionary algorithm strikes a balance between exploration and exploitation, achieving the best performance in the most complex domain of approximating a human face.

The characteristics of our search space also makes our application interesting for optimization research and we see the application of more recent approaches such as

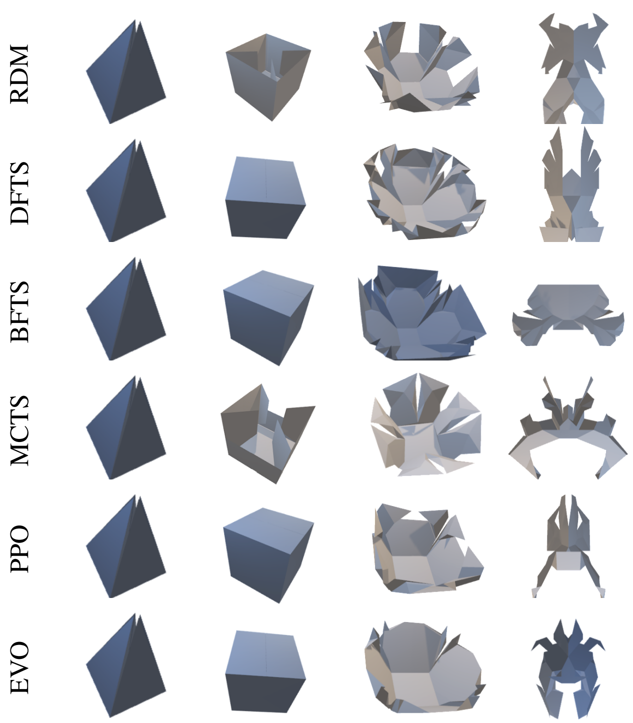

Figure 5: The figure shows the best approximations found for each method over all of the ten runs, with the overall best approximations across all methods highlighted in blue.

GFlowNets [Bengio *et al.*, 2021] as a promising direction to further explore the possibilities here.

4.2 Shape Imagination

With shape imagination we address the second use case of our environment: instead of approximating a given target shape, we utilize the environment for the more abstract generative design tasks. Specifically, we base ourselves on the good performance of the evolutionary algorithm established before and try out the abstract objectives introduced in Section 3.6. We set the board size to a moderate 13×13 and run our evolutionary algorithm for 500 thousand interactions with the environment. Experiment details are given in Appendix G.

The results of the best pattern found are visualized in Figure 6. See also Appendix H for a visualization of the crease patterns and their folding motion. These results show that our setup allows us to find interesting shapes that reflect our specifications while being developable from a flat surface using a single degree of freedom. In the context of furniture, this means that all these objects can be folded flat and stacked for storage or transportation. Moreover, the single degree of freedom makes them easy to fold, as there is a single continuous motion that leads to the folded shape. We see that there is a lot of potential in our setup for customized objectives when the design specifics are left to the optimization.

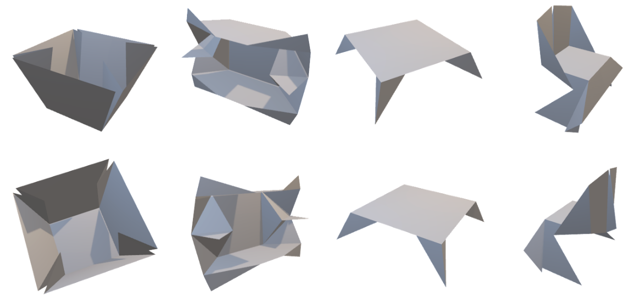

Figure 6: Solutions found on the abstract objectives. From left to right: bucket, shelf, table, chair. The two rows display two viewing angles.

Figure 7: Additional results on the abstract objectives for different symmetry constraints. From left to right: Bucket with only x and y symmetry axes, shelf with only the y symmetry axis, table with only x and y symmetry axes.

4.3 Changing Constraints

As discussed in Section 3.5, if we want to change up the results that we get, besides just changing the random seed used for the search, we can also impose different constraints on the solution space. In the previous section we utilized stricter constraints, for example bucket and table were supposed to be perfectly symmetric, while the shelf was supposed to be symmetric in both x and y axes. In Figure 7 we showcase the results that we get, if these constraints are relaxed. The other settings remain the same (Appendix G), only for the shelf the board size is reduced to 9×9 to counter the search space increase resulting from the removed symmetry.

5 Conclusion

In this paper, we have described an environment for the iterative search of rigidly foldable crease patterns. We discretize the search space to get an efficient model for exploration and constraint imposition. Furthermore, we have demonstrated its utility and limitations for shape approximation by evaluating various search methods, including branch and bound searches, reinforcement learning, and an evolutionary algorithm. We also highlight the potential of our environment for use with customizable objectives and use of constraints to allow the user to guide the search in a desired direction. The proposed rigid origami design environment could prove useful not only in creative endeavors of rigid origami pattern design but also for the application of novel optimization algorithms to origami design and arts.

Contribution Statement

Authors are listed in alphabetical order.

References

[Bengio *et al.*, 2021] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[Callens and Zadpoor, 2018] Sebastien J.P. Callens and Amir A. Zadpoor. From flat sheets to curved geometries: Origami and kirigami approaches. *Materials Today*, 21(3):241–264, 2018.

[Chaslot *et al.*, 2008] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. *AIIDE*, 8:216–217, 2008.

[Demaine and Tachi, 2017] Erik D Demaine and Tomohiro Tachi. Origamizer: A practical algorithm for folding any polyhedron. In *33rd International Symposium on Computational Geometry (SoCG 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[Demaine *et al.*, 2016] Erik D Demaine, Sándor P Fekete, and Robert J Lang. Circle packing for origami design is hard. In *Origami 5: Fifth International Meeting of Origami Science, Mathematics, and Education*, pages 609–626. CRC Press, 2016.

[Dudte *et al.*, 2016] Levi Dudte, Etienne Vouga, Tomohiro Tachi, and Lakshminarayanan Mahadevan. Programming curvature using origami tessellations. *Nature Materials*, 15, 01 2016.

[Hanna *et al.*, 2014] Brandon H Hanna, Jason M Lund, Robert J Lang, Spencer P Magleby, and Larry L Howell. Waterbomb base: a symmetric single-vertex bistable origami mechanism. *Smart Materials and Structures*, 23(9):094009, 2014.

[He and Guest, 2018] Z He and SD Guest. Approximating a target surface with 1-dof rigid origami. In *Origami 7: The Proceedings from the 7th International Meeting on Origami in Science, Mathematics, and Education, Volume 2, Tarquin Publications (2018), 505-520*, volume 2. Tarquin, 2018.

[He and Guest, 2019] Zeyuan He and Simon D Guest. On rigid origami i: piecewise-planar paper with straight-line creases. *Proceedings of the Royal Society A*, 475(2232):20190215, 2019.

[Kawasaki, 1991] Toshikazu Kawasaki. On the relation between mountain-creases and valley-creases of a flat origami. In *Proceedings of the First International Meeting of Origami Science and Technology, 1991*, 1991.

[Kuribayashi *et al.*, 2006] Kaori Kuribayashi, Koichi Tsuchiya, Zhong You, Dacian Tomus, Minoru Umemoto, Takahiro Ito, and Masahiro Sasaki. Self-deployable origami stent grafts as a biomedical application of ni-rich tini shape memory alloy foil. *Materials Science and Engineering: A*, 419(1):131–137, 2006.

[Liang *et al.*, 2018] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

[Maekawa and Kasahara, 1983] Jun Maekawa and Kunihiko Kasahara. *Viva! Origami*. Sanrio, Tokyo, Japan, 1983.

[McAdams and Li, 2014] Daniel A McAdams and Wei Li. A novel method to design and optimize flat-foldable origami structures through a genetic algorithm. *Journal of computing and information science in engineering*, 14(3), 2014.

[Meloni *et al.*, 2021] Marco Meloni, Jianguo Cai, Qian Zhang, Daniel Sang-Hoon Lee, Meng Li, Ruijun Ma, Teo Emilov Parashkevov, and Jian Feng. Engineering origami: A comprehensive review of recent applications, design methods, and tools. *Advanced Science*, page 2000636, 2021.

[Mirhoseini *et al.*, 2021] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[Miura, 1969] Koryo Miura. *Proposition of pseudo-cylindrical concave polyhedral shells*. Institute of space and aeronautical science, University of Tokyo Tokyo, 1969.

[Möller, 1997] Tomas Möller. A fast triangle-triangle intersection test. *Journal of graphics tools*, 2(2):25–30, 1997.

[Morgan *et al.*, 2016] Jessica Morgan, Spencer P Magleby, and Larry L Howell. An approach to designing origami-adapted aerospace mechanisms. *Journal of Mechanical Design*, 138(5), 2016.

[Ng *et al.*, 1999] Andrew Y Ng, Daishi Harada, and Stuart J Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287, 1999.

[Resch, 1970] Ron Resch. The design and analysis of kinematic folded plate systems. In *Proceedings of IASS Symposium on Folded Plates and Prismatic Structures, 1970*, 1970.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016.

[Silver *et al.*, 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[Simm *et al.*, 2020a] Gregor Simm, Robert Pinsler, and José Miguel Hernández-Lobato. Reinforcement learning for molecular design guided by quantum mechanics. In *International Conference on Machine Learning*, pages 8959–8969. PMLR, 2020.

[Simm *et al.*, 2020b] Gregor NC Simm, Robert Pinsler, Gábor Csányi, and José Miguel Hernández-Lobato. Symmetry-aware actor-critic for 3d molecular design. 2020.

[Tachi, 2010] Tomohiro Tachi. Freeform variations of origami. *Journal for Geometry and Graphics*, 14, 01 2010.

[Turner *et al.*, 2016] Nicholas Turner, Bill Goodwine, and Mihir Sen. A review of origami applications in mechanical engineering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 230(14):2345–2362, 2016.

[Vinyals *et al.*, 2019] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, Nov 2019.

[Wang, 2019] Meixin Wang. *An Origami-Based Soft Actuator and the Application as A Soft Gripper*. PhD thesis, The George Washington University, 2019.

[Yoshimura, 1955] Yoshimaru Yoshimura. On the mechanism of buckling of a circular cylindrical shell under axial compression. *NACA Tech. Note No. 1390.*, 1955.

[Zimmermann and Stanković, 2020] Luca Zimmermann and Tino Stanković. Rigid and flat foldability of a degree-four vertex in origami. *Journal of Mechanisms and Robotics*, 12(1):011004, 2020.

[Zimmermann *et al.*, 2020] Luca Zimmermann, Kristina Shea, and Tino Stanković. Conditions for rigid and flat foldability of degree-n vertices in origami. *Journal of Mechanisms and Robotics*, 12(1):011020, 2020.