A Parameterized Algorithm for Flat Folding

David Eppstein

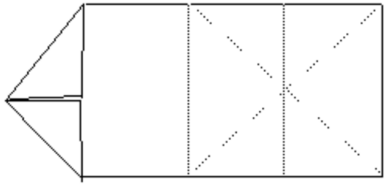
CCCG 2023

From the 1990s world wide web

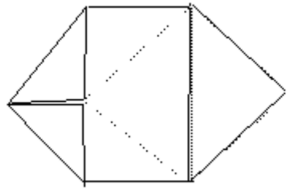
Origami Page

Welcome to the origami page which contains complete and detailed instructions on how to build a simple origami model in 3 easy and one difficult steps. I have entitled this fold "Rabbit style object on geometrical solid".

Step 1:First take a 2:1 rectangular piece off origami paper - ideally rabbit colored on one side and geometric object/rabbit's ear coloured on the other.



Step 2:Fold the corner flaps of the left hand square to the centre and crease the right hand square as shown.



Step 3:Fold the right hand square into this simple base by pinching in the bottom centre crease so that it meets the top centre crease. Crease the left hand square as indicated.

Step 4:Fold all remaining bits of paper on the left hand square into a rabbit-like formation while folding simultaneously the right hand square into a geometric formation.

Et voila, as the Japanese say. The completed rabbit-style object on geometric solid. This photo was taken in my office by pointing the Indy Cam and pressing the button.

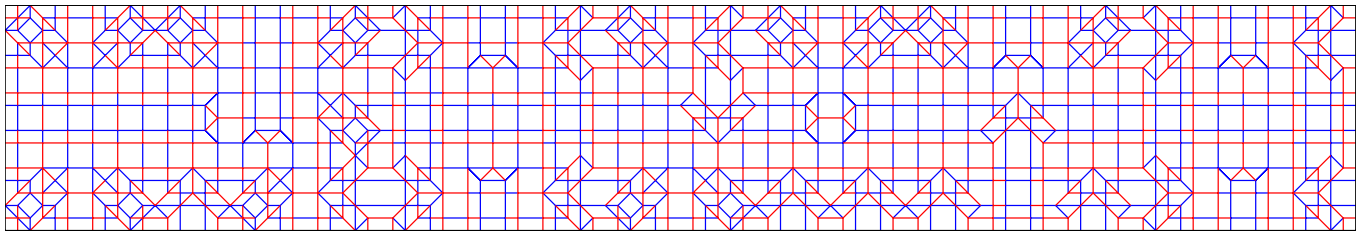
This model was folded by me from the design worked out by Fred Rohm

https://web.archive.org/web/20050816201015/http://www.richardclegg.org/htdocs/origami.html

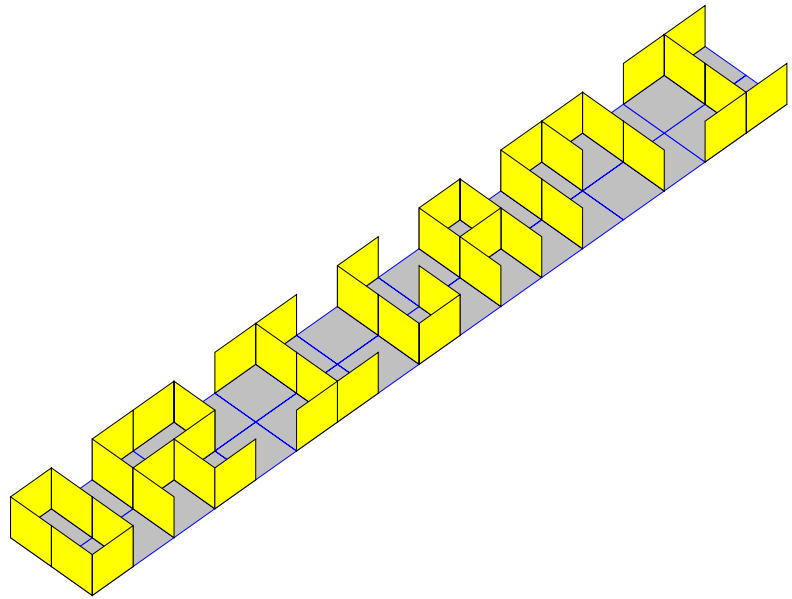
So how hard are origami folding instructions?

We need to formalize this as a computational problem

Input: 2d folding pattern (planar straight line graph of folds), possibly labeled with mountain/valley folds



Output: the folded shape



Simplifying assumption: Shape folds flat in the plane (not like this)

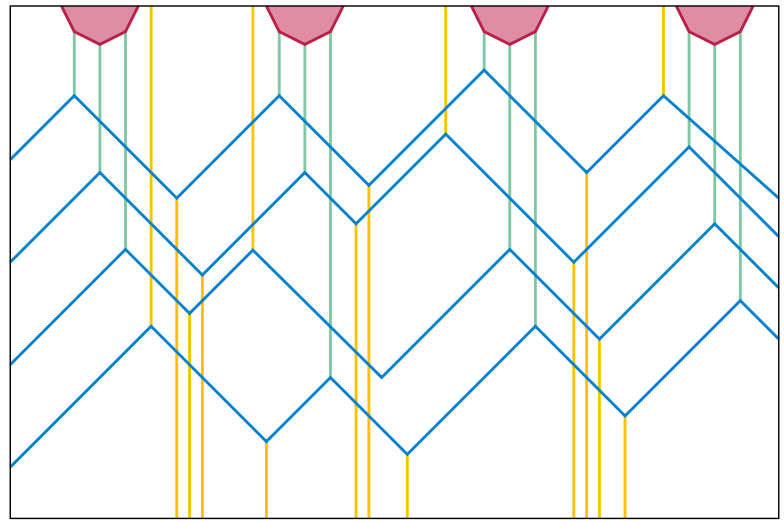
Classical complexity: hard problems $\in$ hard complexity classes

[Bern and Hayes 1996] claim: testing if a pattern can be folded flat (with or without mountain–valley assignment) is NP-complete

Proof sketch: Reduction from not-all-equal-3-sat

Zigzagging pleats (blue) represent variables

More pleats (green) carry data to clause gadgets (red)



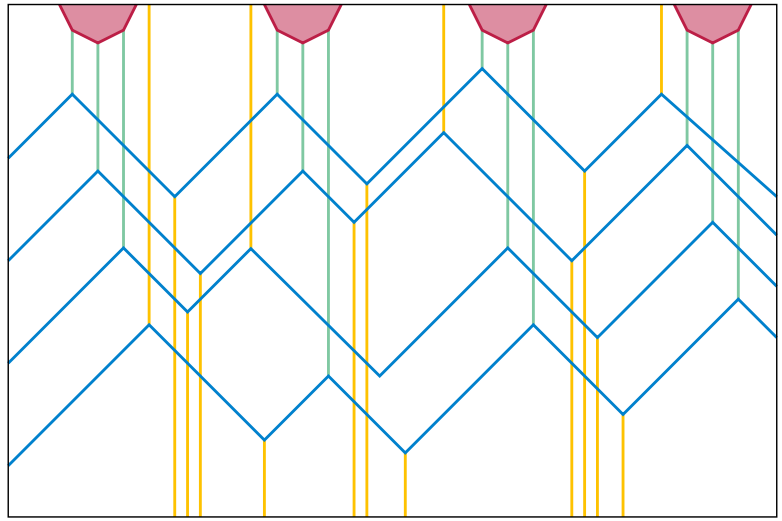
Sadly, their gadgets do not work correctly [Akitaya et al. 2015]

Instead, Akitaya et al. provide new gadgets for similar reduction

(Green connectors extend down as well as up; zigzagging not needed)

Parameterized complexity: Analyze difficulty by more than total input size

Hard inputs from NAESAT have folds along the edges and diagonals of a grid of size $O(\text{clauses} \times \text{variables})$

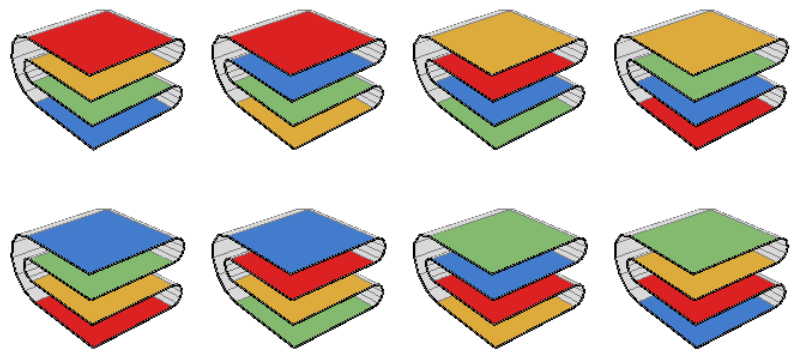


Maybe it's easier to fold patterns in which one of the two grid dimensions is small?

Grid dimension is the wrong parameter

Reason 1: It doesn't apply to non-grid folding patterns

Reason 2: We don't even know how to fold $3 \times n$ square grids

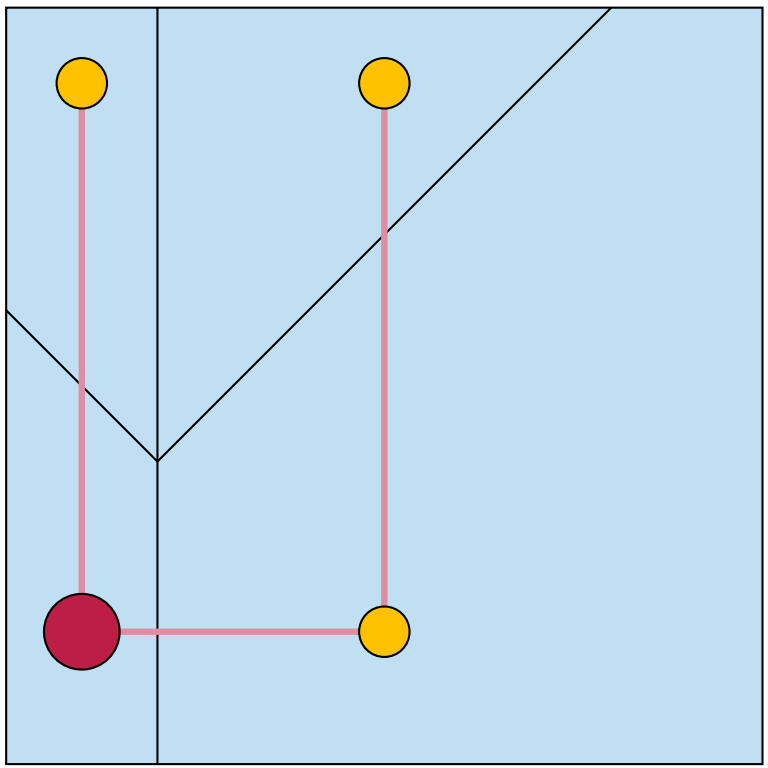


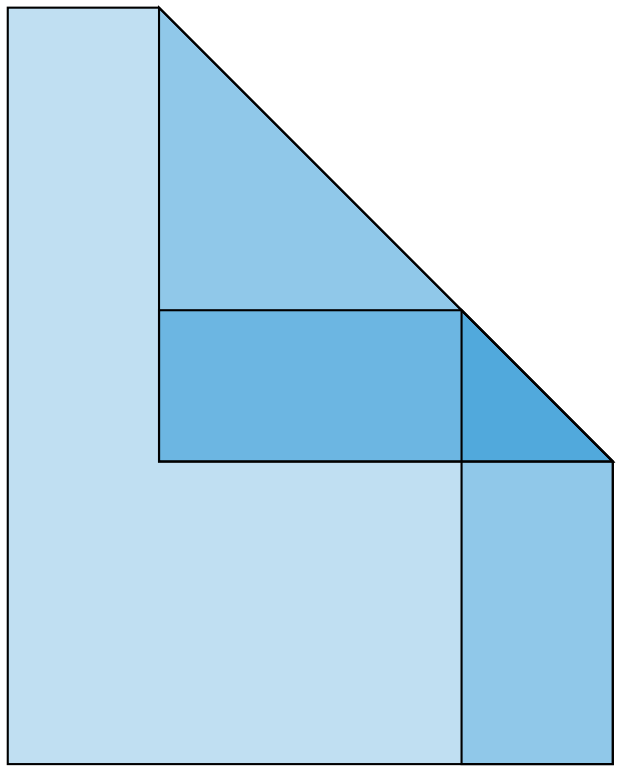
The still-unsolved "map folding problem": flat-fold a grid with a mountain–valley assignment on all grid edges

Positioning the polygons

Observation: It is easy to map each polygon of a folding pattern to its position in the flat-folded result:

- Connect polygons in a (dual) spanning tree
- Fix one polygon in place as tree root
- Reflect the rest across the fold lines along path to root

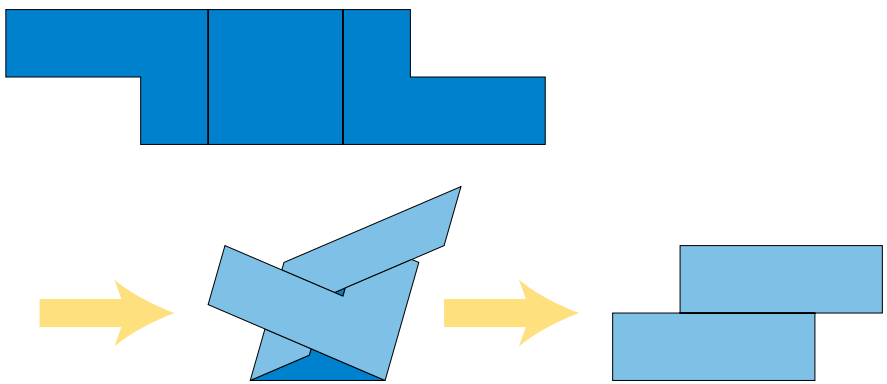




Difficulty: Choosing above-below ordering

Map folding is difficult because the number of possible orderings is factorial in the number of map squares!

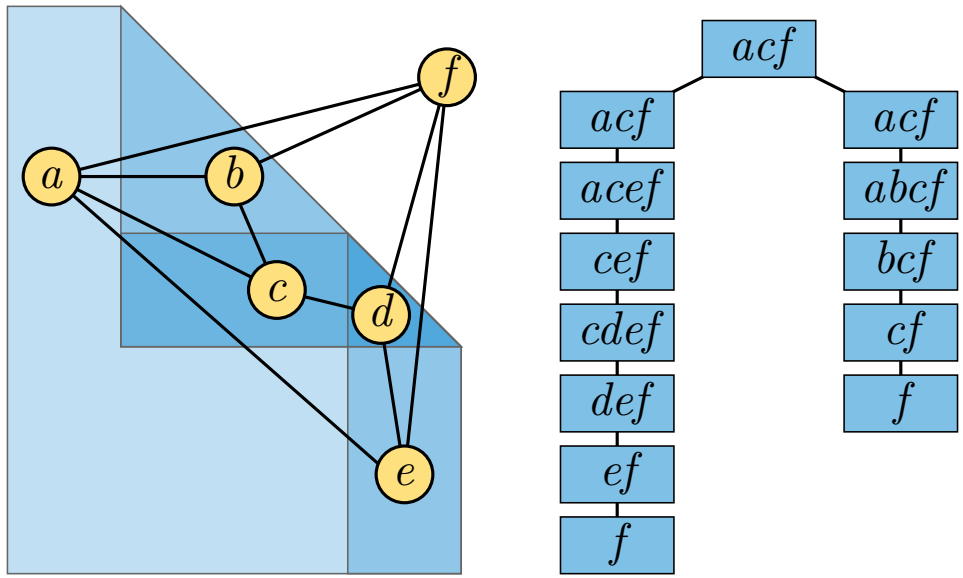
For some folding patterns, the polygons may not even have a consistent ordering



We need to consider a finer subdivision into polygons: subregions of the folded result, not subregions of the folding pattern

Main idea of new algorithm

Make graph with vertex for each subregion of folded result
Edge connecting subregions that share a boundary



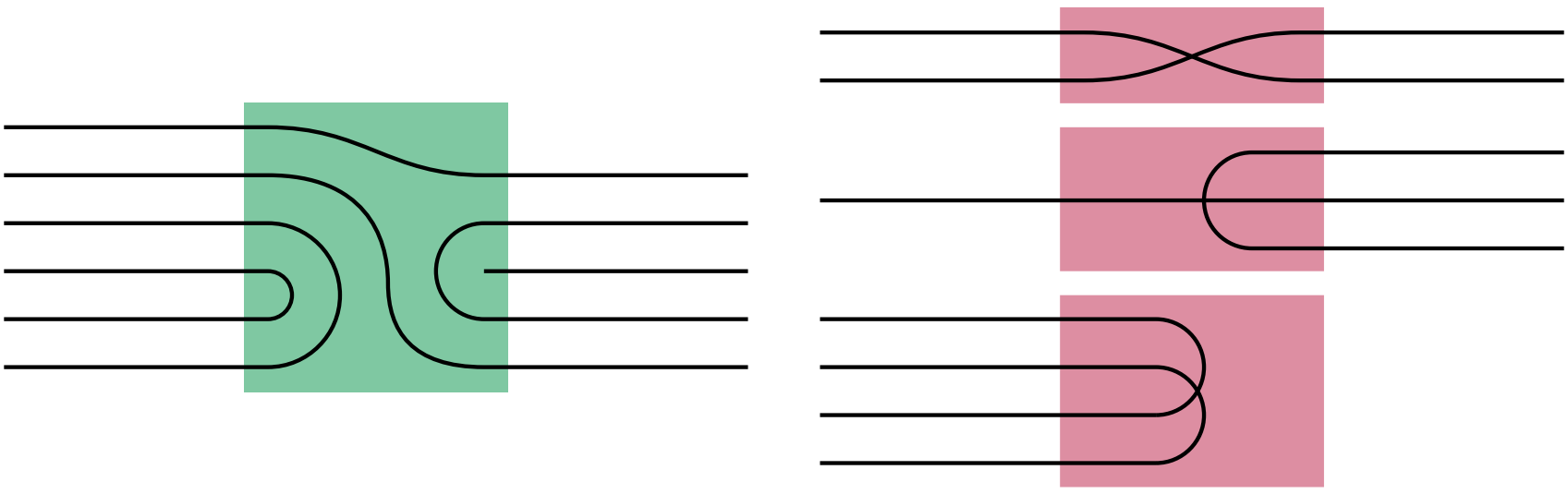
Two parameters: treewidth and ply

(ply = max number of layers, here 3 in region d)

Use dynamic programming to choose layer ordering for each vertex, consistent with orderings of its neighbors

Some details

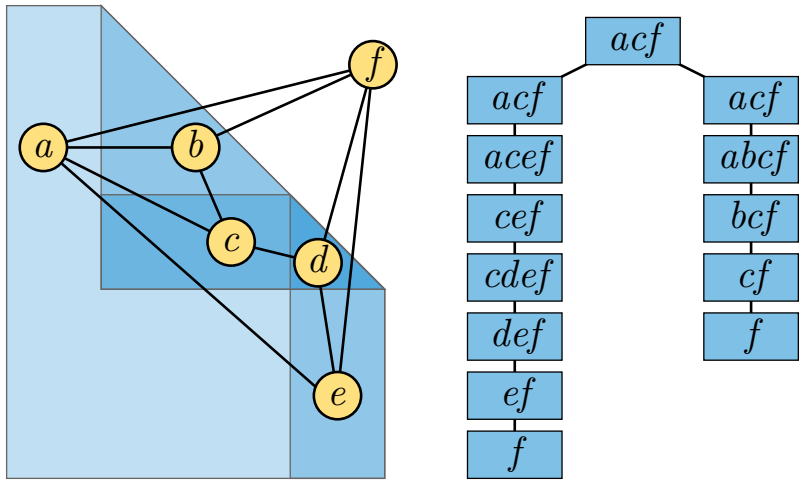
What does it mean for neighboring regions to have consistent orders?



We also need ordering to be consistent within a single region

More details

Dynamic programming algorithm processes rooted tree decomposition in bottom-up order



Each tree node has "bag" of $\leq$ (width + 1) subregions

We compute set of "consistent states":

- Tuple of layer orderings for subregions in bag
- Consistent within each subregion
- Consistent for neighboring subregions of bag
- Comes from consistent states in child bags

Fine-grained complexity: Give me optimality or give me consequences

Time of our algorithm $\approx (\text{ply}!)^{(\text{width}+1)} \cdot \text{creases}^2$

- Polynomial in size of folding pattern
- Singly-exponential in treewidth
- Factorial in ply

Exponential time hypothesis: SAT (and NAESAT) require time c^n

$\Rightarrow$ NAESAT-based folding instances require $c^{\text{grid height}}$, ply $= O(1)$

$\Rightarrow$ our exponential dependence on treewidth is necessary

(faster algorithm would translate back into subexponential SAT)

Still open

How to explain or improve our factorial dependence on ply?

Map folding leads to a one-vertex graph (treewidth = 0)
with high ply (= number of squares)

Is it hard?

Can we quantify its hardness?

References and image credits

Hugo A. Akitaya, Kenneth C. Cheung, Erik D. Demaine, Takashi Horiyama, Thomas Hull, Jason S. Ku, Tomohiro Tachi, and Ryuhei Uehara. Box pleating is hard. In Jin Akiyama, Hiro Ito, Toshinori Sakai, and Yushi Uno, editors, *Discrete and Computational Geometry and Graphs – 18th Japan Conference, JCDCGG 2015, Kyoto, Japan, September 14–16, 2015, Revised Selected Papers*, volume 9943 of *Lecture Notes in Comput. Sci.*, pages 167–179. Springer, 2015. doi: 10.1007/978-3-319-48532-4_15.

Marshall Bern and Barry Hayes. The complexity of flat origami. In *Proc. 7th ACM-SIAM Symposium on Discrete Algorithms (SODA '96)*, pages 175–183, Philadelphia, PA, 1996. Society for Industrial and Applied Mathematics. URL `https://portal.acm.org/citation.cfm?id=313852.313918`.

Robert Dickau. Eight ways to fold a 2x2 map along its creases. CC-BY-SA image, March 24 2010. URL `https://commons.wikimedia.org/wiki/File:MapFoldings-2x2.png`.