Folding Any Orthogonal Maze

Erik D. Demaine* Martin L. Demaine* Jason Ku*

We develop an algorithm to fold an $O(n) \times O(n)$ square of paper into an $n \times n$ orthogonal maze of ridges protruding out of a square base.

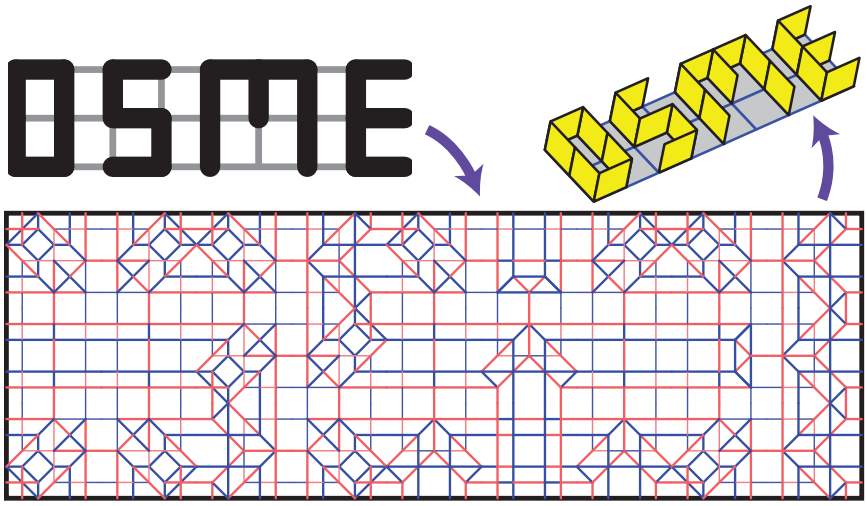

Figure 1: Folding 3D letters from a rectangle of paper. Font design from [Demaine et al. 10]. Mountains are red; valleys are blue; 180° folds are thick; 90° folds are thin.

1 Introduction

In most real-world origami, the final folded model is only a small factor smaller than the original piece of paper. This property is obviously useful for practical folding, but we are far from understanding what makes it possible mathematically. The universality result for origami by [Demaine et al. 00] uses an extremely large scale factor, and the computational

*MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA, {edemaine,mdemaine,jasonku}@mit.edu

origami design techniques of TreeMaker [Lang 03] and Origamizer [Demaine and Tachi 10] solve their nonlinear optimization problems via heuristics and so cannot definitively say which models require large scale factor (though in practice they work well for many models of interest).

In this paper, we prove that a wide family of *origami maze* designs can be folded with a small scale factor. We develop an algorithm to fold a square of paper into any orthogonal maze, consisting of vertical walls protruding equal heights out of a square floor. More precisely, given an orthogonal graph drawn on an $n \times n$ square grid, we fold a $(2h+1)n \times (2h+1)n$ square of paper into the square with the orthogonal graph extruded orthogonally to a specified (uniform) height h. The zero-thickness ridges could form a path like the Hilbert curve, a maze or labyrinth, troughs for liquid distribution, or letters of the alphabet (as in Figure 1).

How good is the scale factor of $2h+1$? The answer depends on the target orthogonal graph. The scale factor always must be at least 1, because the intrinsic diameter of the target shape (even for an empty graph) is at least $\sqrt{2}\,n$, and because intrinsic diameter can only decrease by folding. If the graph is connected and spans all $(n+1)^2$ points, then it has $(n+1)^2 - 1$ edges, so at least one of the $2n$ rows or columns must contain at least $[(n+1)^2 - 1]/(2n) = n/2 + 1$ edges of the graph, which induces an intrinsic straight line of length $n + (n/2+1)(2h) = n(h+1) + 2h$ (going up and down each ridge of height h). The diameter of the paper must be at least this long, proving that the scale factor must be at least $(h+1)/\sqrt{2}$, so our algorithm is guaranteed to be a factor of slightly less than $2\sqrt{2}$ away from optimal. If a row or column has all $n+1$ edges, then our algorithm is definitely within a factor of $\sqrt{2}$ from optimal. For the complete $n \times n$ grid graph, we suspect that our folding is optimal (at least among watertight foldings). We also suspect that our folding is very close to optimal for "most" (e.g., random) subgraphs of the $n \times n$ grid.

A particularly practical situation, used in our examples and implementation, is $h = 1$. In this case, we start with a square just three times larger in dimensions than the final shape. Furthermore, The number of layers of paper that come together at any point is bounded by a constant.

Our foldings are *watertight* [Demaine and Tachi 10]: the boundary of the paper maps to the boundary of the model. In contrast, the original algorithm for folding any polyhedral surface [Demaine et al. 00] is inefficient and not watertight. Origamizer [Demaine and Tachi 10] provides a family of foldings that may include similarly efficient (and watertight) foldings, but does not provide an efficient algorithm to find such a good folding. The box-pleating techniques of [Benbernou et al. 09] are most closely related, but applied in a straightforward matter, would use a square of side length $\Theta(n^2)$.

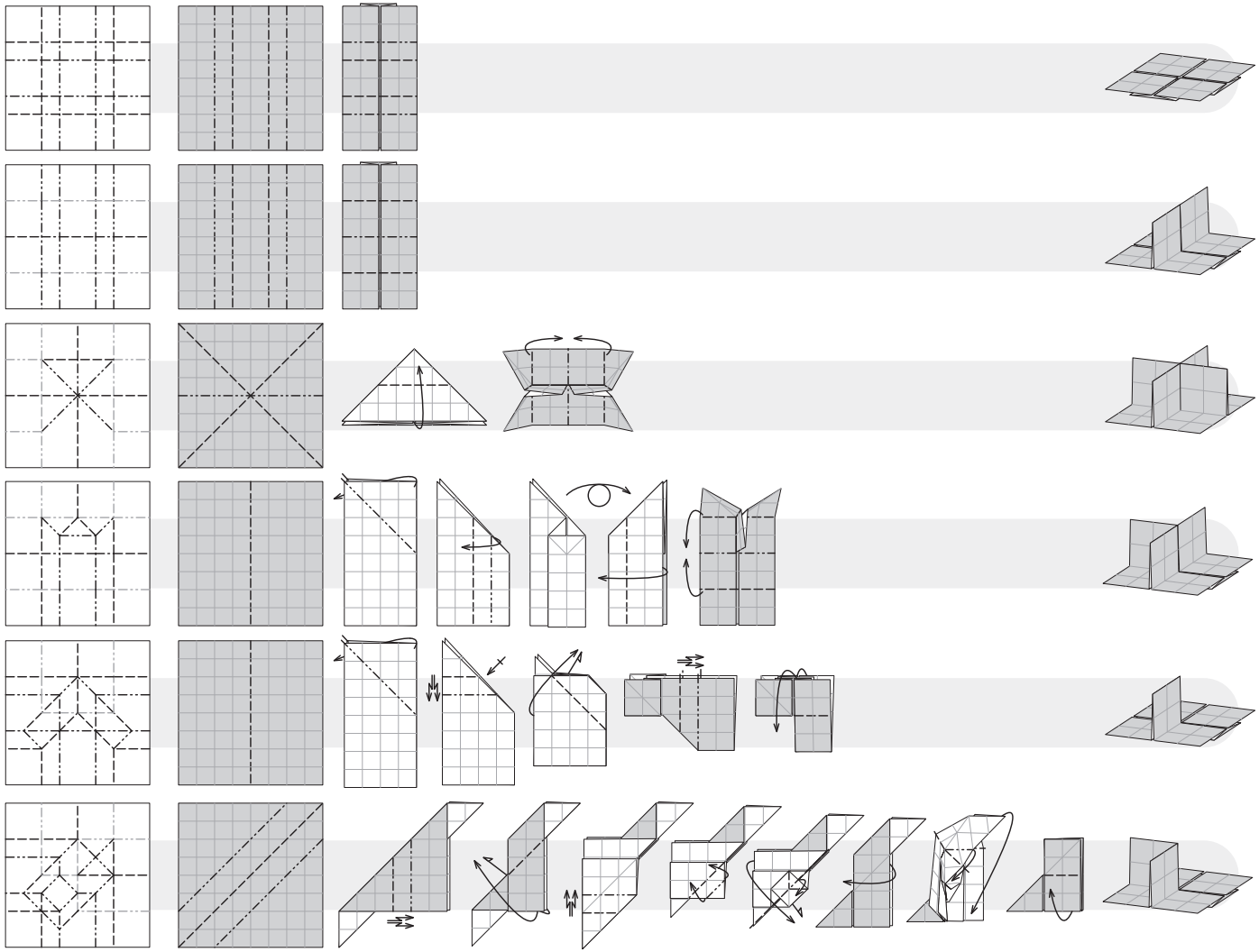

Figure 2: Diagrams establishing the local foldability of each vertex gadget. From top to bottom: degree 0, straight, degree 4, degree 3, degree 1, and corner. Valleys are dashed; mountains are dot-dashed; 90° folds are grey; 180° folds are black.

2 Algorithm

Our algorithm uses an appropriate folding gadget at each grid point, depending on which of the four incident edges should be extruded. In all, there are six gadgets up to rotation, shown in Figure 2. The step-by-step diagrams are not part of the algorithm, but rather serve to describe the layering of the final folded state.

The algorithm tiles the crease pattern and corresponding folded state, for each vertex, in a grid pattern according to the orthogonal graph. Figure 1 shows a simple example. Here we use that all gadgets have a consistent interface on each of their four sides (depending on whether the side is a ridge or floor), allowing gadgets be combined in an arbitrary combination.

Depending on the height h of extrusion, the gadgets in Figure 2 may get placed very close to each other. Consider the 90° corner gadget shown in the middle of Figure 3. The crease pattern extends outside the central 2×2 square reserved for the gadget, in the lower left. If $h > 1$, then the crease pattern may overlap an adjacent gadget, which is invalid. To fix this problem, we thin the excess structure near the floor (nonridge) edges of every gadget by sufficiently many sink folds, as shown on the right of Figure 3. This thinning is necessary only for large extrusion heights h.

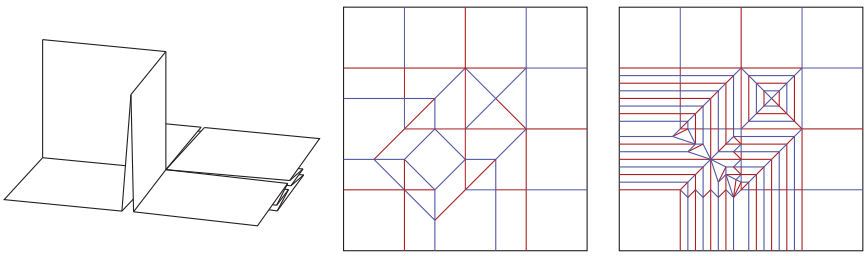

Figure 3: Corner gadget: 3D folded state, simple crease pattern, and thinned crease pattern. Mountains are red; valleys are blue; 180° folds are thick; 90° folds are thin.

The algorithm runs in linear time and is easy to implement. We have implemented the algorithm as a freely available web application:[1] you can design an orthogonal graph or generate a random maze, and the application produces a crease pattern, which you can print and fold into your design. The application assumes that $h = 1$. Also implemented is a simple orthogonal-graph font [Demaine et al. 10] for writing messages such as Figure 1.

[1] http://erikdemaine.org/fonts/maze/

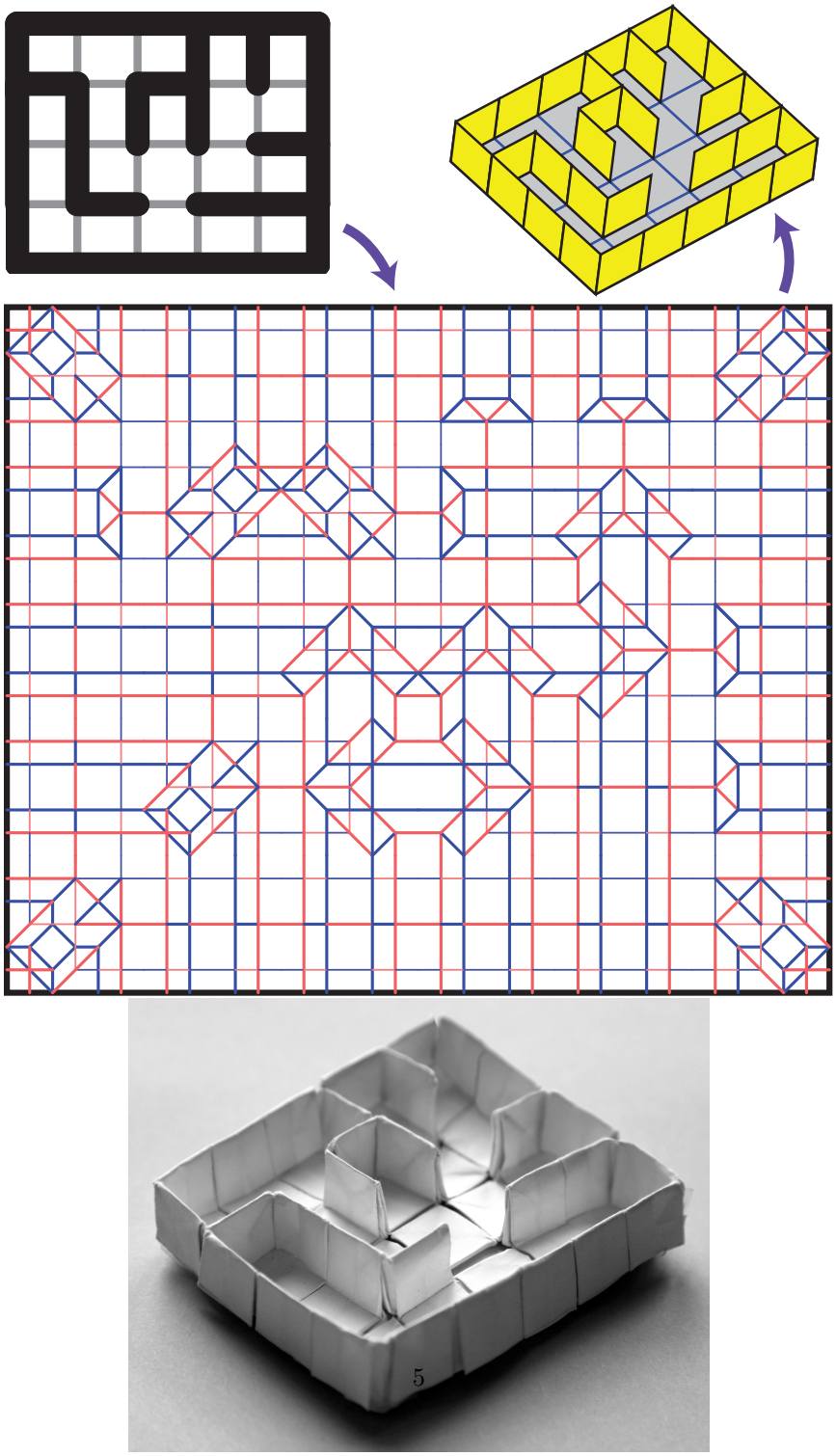

Figure 4: 4×5 origami maze. [Folding by Christopher Chin.]

Acknowledgments

We thank Erez Lieberman for posing an artistic challenge—folding a Hilbert curve—that ended up inspiring this work; see Figure 5. We thank Aviv Ovadya and Tomohiro Tachi for helpful comments.

Figure 5: Folding of a Hilbert curve from a square of paper by Jason Ku, using an early version of the corner and straight gadgets. This photo appeared in *Technology Review* and *Popular Science* (October 8, 2009).

References

[Benbernou et al. 09] Nadia Benbernou, Erik D. Demaine, Martin L. Demaine, and Aviv Ovadya. "A Universal Crease Pattern for Folding Orthogonal Shapes." arXiv:0909.5388, 2009. http://arxiv.org/abs/0909.5388.

[Demaine and Tachi 10] Erik D. Demaine and Tomohiro Tachi. "Origamizer: A Practical Algorithm for Folding Any Polyhedron." Manuscript, 2010.

[Demaine et al. 00] Erik D. Demaine, Martin L. Demaine, and Joseph S. B. Mitchell. "Folding Flat Silhouettes and Wrapping Polyhedral Packages: New Results in Computational Origami." *Computational Geometry: Theory and Applications* 16:1 (2000), 3–21.

[Demaine et al. 10] Erik D. Demaine, Martin L. Demaine, and Jason Ku. "Origami Maze Puzzle Font." In *Proceedings of the 9th Gathering for Gardner*. Atlanta, Georgia, 2010. To appear.

[Lang 03] Robert J. Lang. *Origami Design Secrets: Mathematical Methods for an Ancient Art*. A K Peters, 2003.