



HAL
open science

An example of geometric origami design with benefit of graph enumeration algorithms

David Dureisseix

► **To cite this version:**

David Dureisseix. An example of geometric origami design with benefit of graph enumeration algorithms. 2015. hal-01219814v1

HAL Id: hal-01219814

<https://hal.science/hal-01219814v1>

Preprint submitted on 23 Oct 2015 (v1), last revised 24 May 2017 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial - ShareAlike 4.0 International License

An example of geometric origami design with benefit of graph enumeration algorithms

David Dureisseix

University of Lyon, INSA Lyon (France)

Abstract: This article is concerned with an example of complex planar geometry arising from flat origami challenges. The complexity of solution algorithms is illustrated, depending on the depth of the initial analysis of the problem, starting from brute force enumeration, up to the equivalence to a dedicated problem in graph theory. This leads to algorithms starting from an untractable case on modern computers, up to a run of few seconds on a portable personal computer. This emphasizes the need for a prior analysis by humans before considering the assistance of computers for complex design problems. The graph problem is an enumeration of spanning trees from a grid graph, leading to a coarse scale description of the topology of the paper edge on the flat-folded state.

Keywords: spanning tree enumeration; NP-hard; planar geometry; complexity; folding

1 Introduction

Origami (paperfolding without cutting nor gluing) and especially flat-folded model design, strongly relies on planar geometry. Color-changing technique (using appropriately a paper with one color on a face and a second color on the other face) adds some challenges to the previous model design. The most demanding cases use numerous and alternated color changes on the flat-folded state of the paper, and making a chessboard (or checkered patterns) is among the hardest problems, provided that one adds the constraint of starting from a single square sheet of paper. Indeed, starting from a narrow strip of two-colored paper, or from several separated sheets (a technique known as modular origami), drastically reduces the difficulty of designing a checkered pattern.

Several designs appeared during the past 30 years, see Table 1, and the question of the optimality was settled: to get a $n \times n$ checkered pattern, what is the minimal size of the initial square sheet of paper? In its present general form, this question is still open. Nevertheless, once a design is completed, an upper bound for the optimal initial square sheet size is made available. Some

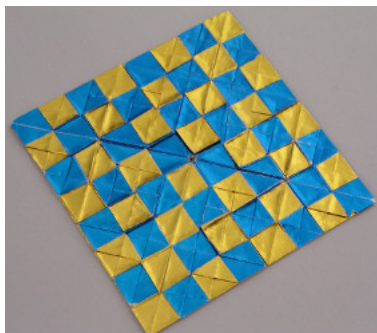


Figure 1: A possibly optimal 8×8 chessboards, from a 32×32 square of paper, after [12].

estimates were also given: with the sensible assumption that a color-change always appears along a side of the initial square sheet of paper, this leads to the question of the length of a continuous path which is followed by the sides of the initial square sheet of paper on the flat-folded state, called the edge diagram. With such an assumption, the answer was that a half-perimeter $s = n^2$ is required [12], Figure 1.

Publication year	Initial paper size	Author
1985	64×64	Hulme [16]
1989	40×40	Casey [4]
1993	36×36	Montroll [24]
1998	40×40	Kirschenbaum [19]
2000	32×32	Dureisseix [12]
2001	32×32	Chen [6]
2007	32×32	Hollebeke [14]

Table 1: Historical designs of some 8×8 chessboards, after [3].

2009 saw a breakthrough. Alleviating the previous constraint on the edge pattern, an asymptotic sharper bound has been given in [8] together with a general constructive proof. The authors found (for n even): $s = \frac{1}{2}n^2 + 8n + 85(n \bmod 4)$. This bound is nevertheless outperforming the previous designs only for $n > 16$, see Figure 2.

Apart from the chessboard problem, another challenge emerged recently on social networks: a design of a pixel matrix [30]. The difference with the pixel project [7] (for which small square modules of two possible different colors are assembled together to form a pixelated image) lies on the use a single sheet of paper. Moreover, as for a LED matrix, each board square should be able to change its color simply (e.g. with a single paper flip) and independently of the others. It appears that the proposed 8×8 pixel-matrix design can be obtained from a rectangular paper (an 8×66 strip is possible, a longer strip renders it

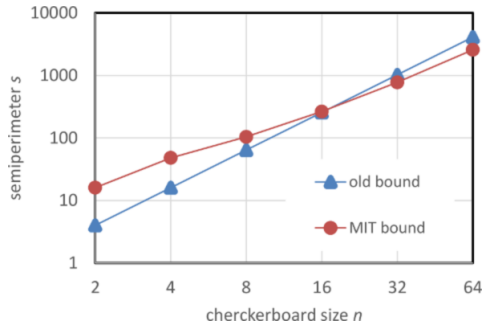


Figure 2: Bounds on complexity for the general problem of the $n \times n$ checkered pattern (n even).

easier; it is not known to the author if a shorter one is feasible). Therefore, the challenge of the design of an optimal 8×8 pixel-matrix chessboard from a square sheet was settled.

This article focus on this last question, together with the possibility of using computers to check the possibilities and to help for designing.

2 Flipping mechanism and optimality challenge

The main argument used herein is to reuse the same edge assumption as for the early chessboard: we still rely on the initial paper edge for the color-change, since it still produces the best known paper optimality, at least for the 8×8 design. Furthermore, a second assumption is needed for the flipping mechanism allowing a color-change on each board square independently; we suggest to design an articulation on the diagonal of the board square. Using a corner or a side of the initial paper square therefore leads to two different folding designs; Figure 3 presents the elementary folding mechanisms to generate such flaps. The folded crease allowing each flap has a length of $\sqrt{2}$ and a perimeter length consumption of 2, both for the corner and for the side mechanisms.

If this flipping mechanism is feasible, it will therefore require the use of 4 corner mechanisms plus $n^2 - 4$ side mechanisms, leading to a semiperimeter with a minimum length of n^2 , that is no more than the straight chessboard. One could therefore challenge that an 8×8 color-changing pixel-matrix can be designed with the same efficiency as for the best 8×8 chessboard, i.e. with a 32×32 square of two-colored paper.

In case of success, this would also exemplify the raise in difficulty by prescribing a design from a *square* sheet of paper. Indeed, the aforementioned strip (8×66 rectangular paper) leads to a mean thickness of the folded model (counted as the average number of superimposed paper layers) to be $(8 \times 66)/(8 \times 8) = 8.25$, while for the square paper, it would raise for the hopefully best case to $(32 \times 32)/(8 \times 8) = 16$.

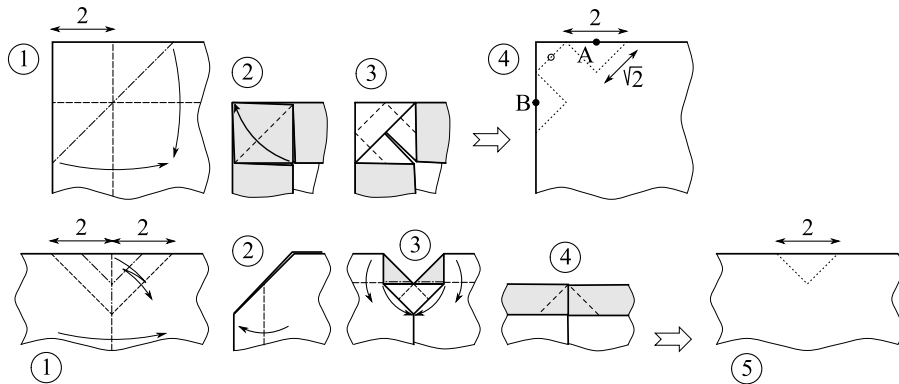


Figure 3: Folding diagram of the flipping mechanisms. Top: corner (1: waterbomb-like folds; 2: fold in half; 3: flip the color of the board squares, unfold), bottom: edge (1: crimp; 2: bookfold; 3: swivel folds; 4: flip, unfold).

3 Design principles for the pixel matrix

To design such a geometric origami model, straight force of computers is not yet sufficient. Indeed, for flat-folding problems, the complexity of the task is very rapidly overwhelming. An underlying basic question concerns the crease pattern, which is the drawing on an unfolded flat sheet of paper of the crease locations as well as their assignments (mountain or valley): given a crease pattern, will it fold flat? This question appeared to be not trivial. Some general conditions can be stated [17, 1] but they are hardly usable in practice; some necessary local constraints around each vertex (i.e. each crease intersection) [15, 10] are nevertheless easier to express, and one could expect relying on computers for running algorithms that could check the foldability. Unfortunately, in their general forms, these problems are hard to solve. For instance, the simple companion problem of crease assignment (given an arbitrary crease pattern, decide for each fold if it is mountain or valley for the model to fold flat) is NP-complete [2]. General case is therefore untractable, though some tools are already available to help designing or checking rigid origami foldability, such as *TreeMaker* [22, 23], using circle packing [9], and *Rigid Origami Simulator* [28, 29], using mechanism theory.

Design searching therefore needs for intermediate steps in the genuine problem, that may lead to more amenable solution strategies. The proposal for the pixel-matrix design is to split the problem into simpler subproblems: after the previous design for a flipping mechanism on the paper edge, one could focus on an edge path determination, and on a fold propagation from the edge to the center of the paper.

3.1 Generalized edge pattern and search complexity

With the previous flipping mechanism, a first subproblem concerns the placement on the flat-folded model (i.e. the $n \times n$ chessboard pattern) of the articulation of the flap for each board square. They should split each board square in half along one of its diagonal, and since they are connected to a paper initial edge as in Figure 3, the set of all those crease locations should (i) be a continuous closed curve (it should possess only one closed loop), (ii) without crossing (but touching is allowed) since it should be on the top surface of the folded model, (iii) passing through one diagonal of each of the n^2 board squares. This path could be named generalized edge pattern in reference to its counterpart for the classical chessboard [12].

As a guide for the design, the paperfolder is therefore interested in selecting such a path that could be mapped onto the edge of the initial square sheet of paper. He could also select it with additional considerations such as symmetries (that may allow to reduced the size of the problem), estimates the difficulty of the task, etc. A useful information is therefore the enumeration of all possible paths. This part of the problem is prone to computerization and is discussed in the following.

3.1.1 Brute force approach

Without a deeper analysis, and since the path splits all the board squares in half, but with two diagonal possibilities each time, a crude enumerating approach consists in selecting one of the two possible diagonals per board square, for all the possible configurations, and testing for the aforementioned constraints. This kind of problem is usually not polynomial in time, since the difficulty lies in the number of possible selections of a set of n^2 diagonals: there are $p_1 = 2^{(n^2)}$ sets to test. This number increases rapidly with the size n ; Table 2 reports the corresponding values; the 8×8 case seems not to be possible to perform this way¹.

n	p_1	p_2	p_3	N	\tilde{N}
2	16	1	1	1	0
4	65 536	16	4	4	0
6	68 719 476 736 ≈ 69 billions	4 096	495	192	
8	18 446 744 073 709 551 616 ≈ 18 billions of billions	16 777 216 ≈ 17 millions	1 307 504 ≈ 1.3 millions	100 352	3 924

Table 2: Number of cases to generate and test, depending on the problem entry size n .

¹The number of cases is close to the solution of the famous ‘wheat and chessboard’ problem

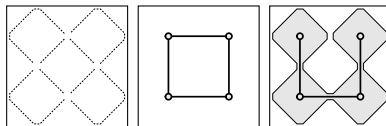


Figure 4: Path location for the generalized edge pattern of the 4×4 pixel-matrix problem, and the corresponding graph. The only solution, but which does not allow the corner placement is also depicted.

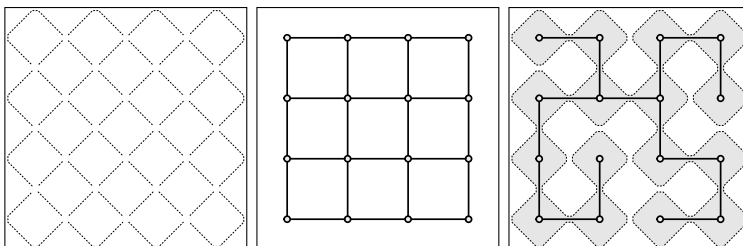


Figure 5: Path location for the generalized edge pattern of the 8×8 pixel-matrix problem, and the corresponding graph. One solution among others is also depicted.

3.1.2 Path growing approach

A second approach consists of making a non-crossing path grows in a continuous way, by step of one diagonal at a time (called a segment in the following) and exploring all alternatives, i.e. the possible different orientations for the next segment. Due to the closed and continuous characters of the path, not all the diagonal sites are feasible, opportunely reducing the problem size: corner board squares could be split in only one way (otherwise there is a pending segment at the corner, preventing the path to be continuous), and for an even n (only considered in the following), the constraint propagates to allow only one feasible diagonal per board square. The possible location for the path is depicted for a 4×4 problem in Figure 4 (left), and for an 8×8 problem in Figure 5 (left). Moreover all the successive segments are connected at right angle. This could be understood as a consequence of the path requirements: if two segments are aligned, the path should also contain a segment going perpendicular at the connection node (to fill all the board edges) see Figure 6; this segment should have a pending end-point (it cannot cross the previous sub-path of the two segments); therefore, the path could not be closed.

All in all, these criteria reduce the problem to the choice between two connections at each missing point, leading to a number of possibilities of $p_2 = 2^{\left(\frac{1}{2}n^2 - n\right)}$. For each, the single-loop constraint should be tested. Though this number still grows rapidly with n , the 8×8 case is now possible to be computerized, see Table 2.

The author programmed an algorithm to solve the path search problem by

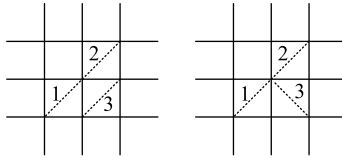


Figure 6: Local connections of the path when segments 1 and 2 are aligned. Left: segment 3 cannot be connected to the previous path of segments 1 and 2 due to its orientation; right: segment 3 with the converse orientation has a pending end point and the path cannot be closed. Conclusion: successive segments 1 and 2 cannot be aligned.

making the path grows, which is a somehow engineering approach to find a way to get a solution. From the $p_2 = 16\,777\,216$ cases, only 20 826 are single-loop non-crossing paths. Among these last ones, some are symmetric to others, in the symmetry group of the square (dihedral group of order 8). Once eliminating them, the number of solutions reduces to 12 600.

3.1.3 Pathway to the graph theory

Due to the closed, continuous and non-crossing characters of the path, it splits the $n \times n$ square domain in an outer and an interior subdomains. These subdomains are composed by cells that are the union of half diagonal parts of the board squares sharing a corner (called vertex in the following), i.e. squares tilted by 45 degrees; Figures 4 and 5 (right) depict these two subdomains in different gray levels. Focusing on the interior gray subdomain, connected by cell corners and whose contour is the searched path, an equivalent graph can be defined. The graph vertices are the aforementioned $\nu = (n/2)^2$ previous vertices, and the arcs relate all the vertices, without loop, containing the connectivity information of the cells. This is therefore a spanning tree of the square grid graph with ν vertices, known to have $\nu - 1$ arcs [31]. The interest of this new problem formulation is to be able to rely on numerous previous works on graphs.

The problem of finding all the possible paths is therefore casted into a spanning tree enumeration. N denotes the number of these spanning trees, and is reported in Table 2. It can be obtained with the Kirchhoff's matrix-tree theorem [18, 5, 11]. For the 4×4 pixel matrix, one gets $N = 4$ (but all are symmetric of the first one — mirroring or rotating — so only 1 spanning tree is interesting, leading to a single solution for the path). This number also increases rapidly with the size of the board: for the 8×8 pixel matrix, one gets $N = 100\,352$. This is a particular integer sequence known as A007341 [27, 25].

A first approach lies in choosing the arcs of a possible spanning tree between those of the grid graph, and testing for each the nodes involved and the acyclicity. The number of graphs to be tested is then $p_3 = C_e^{\nu-1} = \frac{e!}{(e-\nu+1)!(\nu-1)!}$, where $e = n(n/2 - 1)$ is the number of arcs of the grid graph, see Table 2.

This approach is nevertheless not the most efficient. There are several avail-

able algorithms for enumerating all these trees, and their complexity are increasing as $O(N + \nu + e)$. Few efficient stand-alone implementations have been made available, with the notable exception of the `grayspspan` code of D. Knuth [20, 26] that has been used herein. A graph contains essentially a topological information, but with the node coordinates for the problem under concern, a geometrical information is available, and allows to detect the trees that are symmetric, as previously. Once eliminating them, the number of solutions reduces from $N = 100\,352$ to $12\,600$ (indeed, the same as for the previous approach, which is a good cross-check of the computer implementations).

Traducing the problem in term of graphs has two main effects: first it restate the problem as a more generic one for which more efficient algorithms are available; second it reduces its size since the graph is built on a coarser grid than the path of the paper edge. This second feature is kind of a multiscale problem as it can be seen on Figure 5 (right). The involved scales could be quantified by the lengths of the tree and path: for the spanning tree, the length of its $\nu - 1$ edges is $2(\nu - 1)$ while the length of the generalized edge path is $\sqrt{2}n^2$. The length difference is related to the microscale so that the characteristic scale ratio is $(n^2 - \sqrt{2}(\nu - 1))/n^2 \geq 1 - \sqrt{2}/4 \approx 0.646$ (the asymptotic value for large n). The scales are therefore hardly separable. It would certainly be interesting to be able to find even coarser models to reduce further the size of the problem, but such upscalings are somehow case dependent, and not easy to derive. Once a macroscale spanning tree is found, the unique associated path should be built as a microscale corrugation.

3.2 Corner placement constraint

A notable difference between the chessboard design [12] and the pixel-matrix design lies in a strong additional constraint, concerning the corners of the initial square sheet of paper: while the generalized edge pattern of Figure 7 (left) is feasible, this is not the case for the one of Figure 7 (right). Indeed, the distance between points A and B on the flat-unfolded state Figure 3 (top, step 4) is $d = 2\sqrt{2} \approx 2.83$, while on the flat-folded state of Figure 7 (right) it would be $d' = \sqrt{10} \approx 3.16$. However, the paper inextensibility constraint requires $d \geq d'$, and since it is not the case, the generalized edge pattern of Figure 7 (right) is not feasible for a corner of the initial square of paper.

After building all the non-crossing one-loop continuous path of length n^2 from the spanning trees, one should check if they satisfy the corner placement constraint: there should be at least 4 segments equally spaced on the path whose predecessor and successor are not identically oriented (they may be aligned, but should not have the same orientation, Figure 7).

For the 8×8 pixel matrix, among the $N = 12\,600$ paths, this check should be made to eliminate those that do not possess a feasible corner placement. Doing so, only $\tilde{N} = 3\,924$ paths remain (hence the strong constraint prescribed by the corner placement), with 26 ones possessing a vertical or horizontal self-symmetry (none have any other self-symmetry). One can also note that for the 4×4 pixel matrix of Figure 4, the only feasible path does not have any

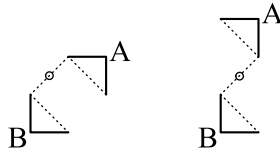


Figure 7: Generalized edge pattern around a corner. Left: feasible path; right: unfeasible path.

possible corner placement; consequently the complexity for the semiperimeter is probably greater than $s = 4^2$ in this case.

For the 8×8 pixel matrix, the 26 symmetric paths are depicted in Figure 8, together with the different possible corner placements. It is interesting to note that there are each time 2 possible corner placements, each symmetric to the other, but that none have a single self-symmetric corner placement. As a consequence, and unlikely the straight chessboard case, the problem cannot be reduced by symmetry to part of the board.

3.3 Onion skin strategy for compatible fold propagation

Once a generalized edge pattern has been selected, the last part of the problem is to fold flat a square sheet of paper that maps its edges (and corners) on the pattern. This part is more difficult to formulate in a way that is easily computerizable.

The more advanced proposal up to the author knowledge is a systematic search on what could be called a ‘lattice origami’ pattern [21]. It assumes all the creases lying on a regular pattern on the initial square sheet of paper, consisting of vertical and horizontal creases distant of a unit value, and on the $\pm 45^\circ$ creases diagonalizing all the previous grid squares. They are depicted in Figure 9. Such a crease lattice is kind of a discretization of the possible crease pattern family and so, it reduces the problem size and makes it more suitable for a discrete treatment by a computer. Unfortunately, this problem is still too computational demanding to be solved by brute force computing.

This part of the problem will therefore rely on the skills and intuition of the paperfolder. Indeed, the problem is at least two-fold: (i) the edge of the paper should map the selected pattern, and (ii) the tortuosity of this path leads to a 2-scale corrugation on the edge of the paper, that have to be propagated from the edge up to the center of the paper, while keeping the folded state flat.

In addition to using the lattice method, we propose to proceed by dealing with successive sub-problems. We propose to cut the paper square in several parts that will have to be re-assembled at the end. If successful, this will lead to a prototype for the model design. The parts are herein successive nested rings lying on perimeters of decreasing sizes, Figure 10. The successive problems consist in flat-folding a part, while mapping its perimeter to the internal side of the previously assembled part. Since the perimeter is decreasing, the size of the

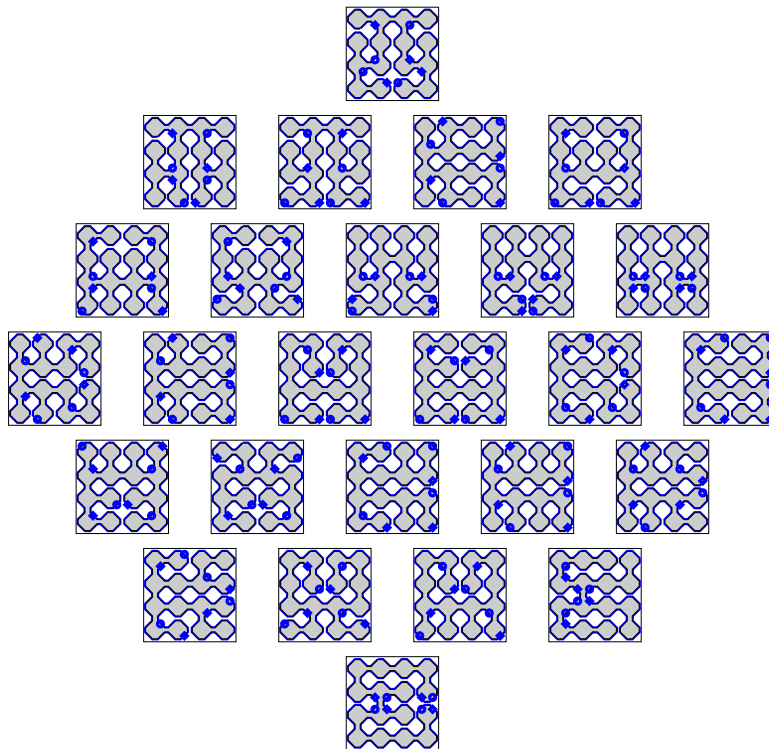


Figure 8: The 26 self-symmetric paths and their corner placement possibilities (denoted by a set of 4 marks along the path).



Figure 9: Pre-creasing a 32×32 square paper to make a lattice of creases.

problem reduces, but a folding compatibility should be satisfied to propagate the folds to the next part to be assembled. If and when the last central square is successfully assembled, the prototype is complete.

Selection of the width of the strips is an interesting issue. The goal is two-fold: each strip should map on the previous edge pattern, and flat-foldability has to be ensured. A thin strip would renders its flat-foldability easy, but do not solve enough foldability compatibility constraints, propagating too much the difficulty to the next inner strip, concentrating progressively the difficulty. Using a large strip allows to solve more progressively this issue but is more difficult to map on the required edge pattern. There is therefore a compromise to be found by choosing accordingly the onion skin width. For the 8×8 pixel matrix problem, a suited choice seems to be the one of Figure 10 (right), with a width of 4. The problem is solved once the central 8×8 square has been successfully mapped to the last perimeter.

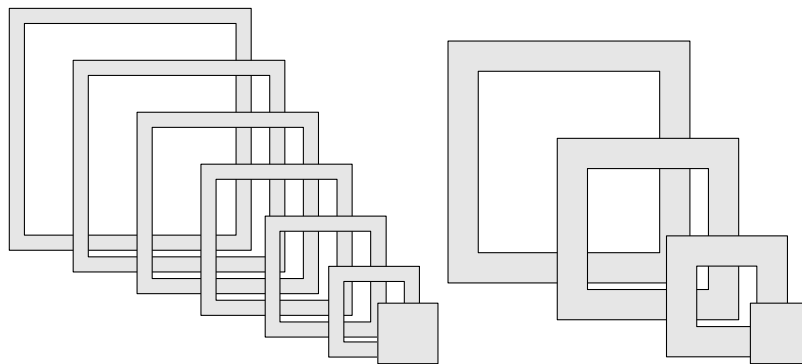


Figure 10: Successive sub-problems with decreasing perimeters, for two choices of width.

A successful prototype is reproduced in Figure 11, allowing to conclude that the 8×8 pixel matrix has the same bound on complexity as the chessboard.



Figure 11: An example of complete prototype.

4 Conclusions

This paper sets the pixel-matrix challenge for the point of view of folding optimality, i.e. constraining the design of the given origami model by prescribing an initial paper sheet to be square. This appears to be a strong constraint that drives the complexity of the task. For a checkered $n \times n$ pattern design without transformable color-change (a static pattern), a bound on the semi-perimeter of the initial square of paper was $s = n^2$, having been improved only for n larger than 16. The pixel matrix adds a complex feature of a flipping mechanism, allowing the local color-change independently on each board square. Though adding a somehow significant difficulty on the design (traded here by a complex corner placement), a surprising result leads to the same complexity for the 8×8 design, as for the chessboard.

Concerning the design process, it has been exemplified that the brute force of computers is still unable to tackle problems we wish to solve in flat geometrical origami. The main issue remains the ability for the user to express the considered problem in a suited form, and especially to split it in several well-chosen consecutive sub-problems, some of them being susceptible to be computerized. In any case, a deeper analysis is required to reduce the problem size, and express the problem within known scientific fields (such as graph theory). For the current example, several analysis with increasing depths lead to a succession of problem size reductions. Then the computer can be helpful to provide information on the potential designs. Nevertheless, several steps still hold on intuition and skills of human paperfolder to complete the design. In this sense, the computer may help, but cannot be substituted to the human for this task. This example illustrates the fact that the dramatic announcement of predominance of computer science on employability [13] could certainly emphasize the complementarity and gain to derive human-computer cooperation, but computers will lack to be autonomous in solving problems that are ill-posed. Certainly the human-computer interface will be a key issue, and problems need to be translated in computer tackling world. Such issues are also discussed about computer science education and learning programs, pros and cons are discussed and currently debated on the utility of early coding courses and/or on recasting traditional courses with a numeric culture orientation, as well as on addressing the issue of the tutor education.

References

- [1] Z. Abel, J. Cantarella, E. D. Demaine, D. Eppstein, T. C. Hull, J. S. Ku, R. J. Lang, and T. Tachi. Rigid origami vertices: Conditions and forcing sets. *arXiv preprint*, 2015. [arXiv:1507.01644v1](https://arxiv.org/abs/1507.01644v1) [math.MG].
- [2] M. Bern and B. Hayes. The complexity of flat origami. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 175–183, Philadelphia, PA, USA, 1996.

- [3] P. Budai. Chequered patterns. <http://www.budaiorigami.hu/en/chequered>. Accessed: 2015-08-29.
- [4] S. Casey. Chessboard. In *West Coast Origami Guild*, volume 19, pages 3–12, 1989.
- [5] S. Chaiken and D. J. Kleitman. Matrix tree theorems. *Journal of Combinatorial Theory, Series A*, 24(3):377–381, 1978.
- [6] S. Y. Chen. Checkerboard. In *Annual OUSA Convention*, pages 72–75, 2001.
- [7] childofsai. Pixel project at BOS 40th Anniversary Convention 2007 (Cambridge). <https://www.flickr.com/photos/childofsai/1435876202>, 2007. Accessed: 2015-08-29.
- [8] E. D. Demaine, M. L. Demaine, G. Konjevod, and R. J. Lang. Folding a better checkerboard. In Y. Dong, D.-Z. Du, and O. Ibarra, editors, *Algorithms and Computation*, volume 5878 of *Lecture Notes in Computer Science*, pages 1074–1083. Springer Berlin Heidelberg, 2009.
- [9] E. D. Demaine, S. P. Fekete, and R. J. Lang. Circle packing for origami design is hard. *arXiv preprint*, 2010. [arXiv:1008.1224](https://arxiv.org/abs/1008.1224) [cs.CG].
- [10] E. D. Demaine and J. O’Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, New York, NY, USA, 2008.
- [11] M. Desjarlais and R. Molina. Counting spanning trees in grid graphs. In *Congressus Numerantium – 31st Southeastern international conference; Combinatorics, graph theory and computing*, volume 145, pages 177–185, 2000.
- [12] D. Dureisseix. Chessboard. *British Origami*, 201:20–24, 2000.
- [13] C. B. Frey and M. A. Osborne. The future of employment: how susceptible are jobs to computerisation. *Retrieved September, 7:2013*, 2013.
- [14] G. Hollebeke. Échiquier. *Le Pli*, 107-108:8–13, 2007. In French.
- [15] T. Hull. The combinatorics of flat folds: a survey. In T. Hull, editor, *Origami 3 – Proceedings of the Third International Meeting of Origami Science, Mathematics, and Education*, pages 29–37. A K Peters, 2002. Also on [arXiv:1307.1065v1](https://arxiv.org/abs/1307.1065v1) [math.MG].
- [16] M. Hulme. Chess sets. In *BOS Booklet*, number 7. BOS, 1985.
- [17] J. Justin. Towards a mathematical theory of origami. In K. M. et al, editor, *Origami Science and Art: Proceedings of the Second International Meeting of Origami and Scientific Origami*, pages 15–29, 1997.

- [18] G. Kirchhoff. Über die auflösung der gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer ströme geführt wird. *Poggendorfs Annalen für der Physik und der Chemie*, 72:497–508, 1847. English translation by J. B. O’Toole, On the solution of the equations obtained from the investigation of the linear distribution of galvanic currents, IRE Trans. on Circuit Theory, CT-5:4-7, 1958.
- [19] M. Kirschenbaum. Chessboard. *The Paper*, 61:24–30, 1998.
- [20] D. E. Knuth. *The Stanford GraphBase, A platform for combinatorial computing*. ACM Press, 1993.
- [21] G. Konjevod. Integer programming models for flat origami. In R. J. Lang, editor, *Origami 4 – Fourth International Meeting of Origami Science, Mathematics and Education*, pages 207–216. A K Peters, 2009.
- [22] R. J. Lang. TreeMaker. <http://www.langorigami.com/science/computational/treemaker/treemaker.php>. Accessed: 2015-08-29.
- [23] R. J. Lang. A computational algorithm for origami design. In *Computational Geometry: 12th Annual ACM Symposium*, pages 98–105, Philadelphia, Pennsylvania, 1996.
- [24] J. Montroll. *Origami inside-out*. Dover, 1993.
- [25] OEIS Foundation Inc. The on-line encyclopedia of integer sequences. Number of spanning trees in $n \times n$ grid. <http://oeis.org/A007341>, 2011. Accessed: 2015-08-29.
- [26] SGB. The Stanford GraphBase. <http://www3.cs.stonybrook.edu/~algorithm/implement/graphbase/implement.shtml>. Accessed: 2015-08-29.
- [27] N. J. A. Sloane and S. Plouffe. *The Encyclopedia of Integer Sequences*. Academic Press, 1995.
- [28] T. Tachi. Rigid Origami Simulator. <http://www.tsg.ne.jp/TT/software>. Accessed: 2015-08-29.
- [29] T. Tachi. Simulation of rigid origami. In *Fourth International Conference on Origami in Science, Mathematics, and Education (4OSME)*, pages 175–187, 2009.
- [30] H. Tahir. Origami dot-matrix. <https://www.flickr.com/photos/31897685@N06/16726050538>. Accessed: 2015-08-29.
- [31] W. T. Tutte. *Graph Theory*. Cambridge University Press, 2001.