Mathematical Methods in Origami Design

Robert J. Lang
Langorigami.com
Alamo, California, USA
robert@langorigami.com
http://www.langorigami.com

Abstract

The marriage of art and mathematics has been widespread and productive, but almost nowhere more productive than in the world of origami. In this paper I will discuss how mathematical ideas led to the development of powerful tools for origami design and will present a step-by-step illustration of the design and realization of a representational origami figure using mathematical design algorithms. Along the way, I will discuss how these mathematical concepts have led to new levels of creative expression within this art.

1 Introduction

The marriage of art and mathematics has been fruitful and productive. Mathematical concepts inspire, enable, and enrich artistic forms, and in many cases, mathematical concepts lead to entirely new expressions, visual, auditory, and conceptual, outside of the traditional milieu of art. It is relatively rare, though, that mathematical ideas have led to a revolution within an existing art—that is, enabling artists to do what they wanted to do within their art, but were previously unable to accomplish. That exact thing is what has happened within the art of *origami*, the Japanese art of paper-folding (usually from an uncut square), over the last twenty years or so. Beginning in the late 1970s and early 1980s, and accelerating more-or-less continuously since then, mathematical design techniques have become essential elements of the origami designers' toolkit, allowing the creation of works only dreamed of in the past, enabling newer, richer, more diverse forms of artistic expression.

Now, while it is tempting to present the world of mathematical origami design as a simple, linear thread of development, in fact there were roots stretching back into the 1950s and 1960s and several different approaches to mathematical design in the 1970s and 1980s investigated by origami artists such as Peter Engel, Jun Maekawa, and Fumiaki Kawahata, among others. But I will talk about the approach I am personally most familiar with, and with which I was closely involved in the development of: circle-river packing. This was, I believe, the first systematic design method that could be used to create and design arbitrarily complicated origami structures.

As often happens in mathematics, when the time is ripe, the same ideas can occur to multiple people, and this was the case with circle-river packing: I (in the US) [1, 3, 4, 6] and Toshiyuki Meguro (in Japan) [10, 9] both developed the concept of circle packing in the late 1980s and then met and compared notes and ideas beginning in 1992, finding surprising overlap between our approaches and sharing ideas in the years that followed. (Shades of Newton and Leibniz, with two differences: Meguro and I got on quite well, and we're only talking about paper-folding, not calculus!) In the years following our meeting, we both traded and refined our concepts and were joined by others who built on and expanded the basic ideas into a regular suite of design algorithms. (I should mention specifically that in recent years I have greatly enjoyed collaborating with Professor Erik Demaine of MIT on the development of improved algorithms and expanding their underlying rigor.)

Fundamental to my own motivation was the goal of developing truly useful techniques for design, and for several years I designed ever more complex origami figures using circles and spacers (the latter were my precursors to "rivers," see below) and nothing more technological than pencil and paper. (My colleague Meguro added compass and straightedge to his crease pattern design arsenal.) However, a paradigm that framed my ideas early on was the idea that it might eventually be possible to use computer programming to solve for complex origami figures, and I began sketching out algorithms for this in the late 1980s. (Early discovery: FORTRAN 77 is a lousy language for computational origami.) Beginning in the 1990s, I began writing what became my program *TreeMaker* (in Object Pascal, then C++), a design program that served both as a repository of algorithms as I developed them and a way to probe their validity and limits [8]. By 1998, *TreeMaker* was up to version 4.0, and had become a useful tool rather than just a toy program; in fact, I realized that it was now capable of solving for origami design patterns that were beyond the ability of an individual armed with only pencil, paper, and compass.

Which is not to say that pencil-and-paper origami designers have not been creating designs as complex as those that *TreeMaker* could create; on the contrary, by recognizing fortuitious geometric alignments and patterns, origami artists could, and still do, design with pencil and paper extremely complex and beautiful works. The most complex origami designs today are usually human-designed, not computer-designed. While its crease pattern may be drawn on computer, advanced origami is designed using a mixture of mathematical ideas and artistic inspiration, with the designer's own imagination the most important tool.

Still, a computerized design serves a useful pedagogical purpose: it provides a vivid illustration of the power of mathematical ideas in achieving a distinctly non-mathematical result. And so, in the bulk of this paper, I would like to walk the reader through the process of concept, design, and folding, for an origami figure: an arthropod. The greatest successes of the circle-river method of design came in the design of insects, spiders, and their ilk, and so I will present here one of my first *TreeMaker* designs: my "Scorpion, opus 379" [5]. Although I have designed more complex figures since then, this remains one of my favorites, both for the geometric structure of its design and its overall appearance.

2 Design

The journey begins with the subject and its abstract description: a tree graph (the "tree" in *TreeMaker*). Or, more prosaically, a "stick figure." The stick figure captures the information that will be produced by the design algorithm, and so its properties—the number, lengths, and connectivities of its edges—are the choice of the designer. The circle-river method of origami design is a step in the creation of a folded figure, but it does not seek to create a specific 3-D form; rather, it produces a shape that has "enough material in the right places." Specifically, it produces a folded shape that has a flap of paper for every appendage of the subject. The algorithm gives relatively little control over the width of those flaps, but it lets one specify the number of flaps, length of each flap, and how they are connected to one another, and for some subjects, that is very useful indeed. That information can be described concisely by a stick figure—which, using the terminology of graph theory, is an "edge-weighted tree graph." Hence, the tree is the starting point for a *TreeMaker* design.

The simplest way to construct the tree is to literally overlay the stick figure on a photograph of the subject, as illustrated in Figure 1. Each stick represents an appendage or body segment of the subject, and will be represented in the origami figure by a flap of paper. The folded shape will be a collection of flaps; we call this collection a "base." Each edge of the tree graph must be assigned a length, which will be the length of the associated flap of paper. By measuring lengths of the various body parts in the photograph, one can choose the desired lengths of the flaps, although it must be noted that one might commonly adjust desired flap lengths from their literal dimensions for artistic reasons.

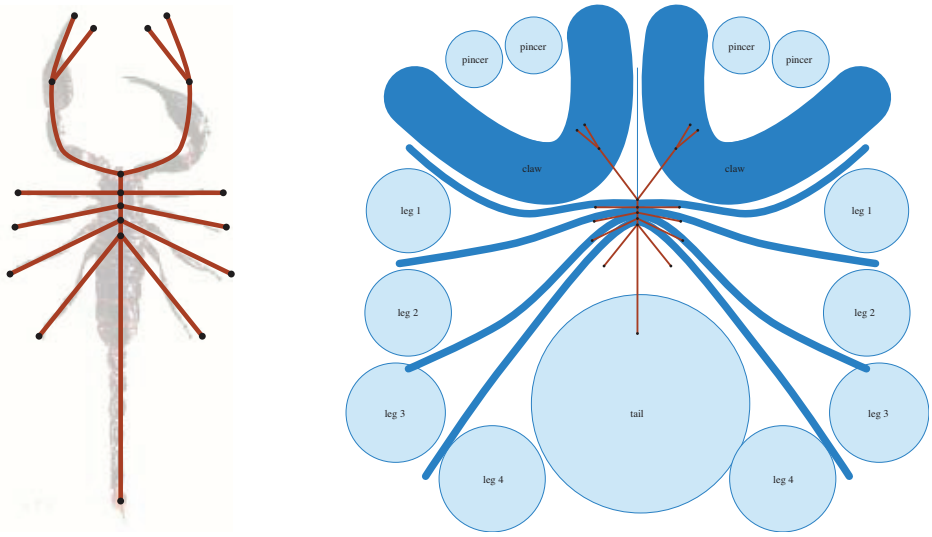

Figure 1 : *Left: the stick figure superimposed over a photograph of the subject. Right: representation of the stick figure by circles and rivers.*

We then transform the stick figure into geometric shapes that represent regions of the paper required by each flap. These shapes are the light and dark regions in the figure. Flaps that are loose at one end, like the legs or pincers, are represented by circles (light), whose radius is equal to the length of the flap. Flaps that are connected to other flaps at both ends, like the body segments or the "arms," are represented by constant-width curves called "rivers" (dark); the width of the river is equal to the length of the corresponding flap. Each geometric shape—circle or river—represents the *minimum* amount of paper required for each of the flaps. It is a remarkable fact that for any valid arrangement of circles and rivers that specify the minimum paper needed, there exists a crease pattern that works for that exact arrangement of circles and rivers.

And so, we must find the most tightly packed arrangement of the circles and rivers within a square. The major rules that apply to this packing are:

- the circles and rivers must be as large as possible but maintain their same relative size to one another;
- they may not overlap;
- the circles do not have to be wholly inside the square, but their centers must be within the square;
- the incidences between the various circles and rivers must match the incidences of their corresponding edges in the tree graph.

The larger the origami base is for a given size square, the more efficient it is; the fewer the layers in each flap; and, generally, the easier it is to fold. And so the first stage of origami design consists of an optimization: finding the most efficient packing of the circles and rivers. This can be done by hand (and it often is), but for this particular design, it is much faster to use *TreeMaker*, which has a nonlinear constrained optimization solver built in. (In the late 1990s, this process took seconds to minutes; with the 1000-fold increase in computing speed, it is now virtually instanteous.) To use *TreeMaker*, we start by drawing the stick figure, as shown in Figure 2 and then typing in the desired lengths of each flap, shown as decimal numbers next to the lines of the stick figure.

The next step is to find the optimum packing of circles and rivers so that all of the circle centers wind up inside the square. *TreeMaker* does this in a fraction of a second, giving the circle arrangement shown in Figure 2. (The rivers are not displayed by *TreeMaker*, but the reader can perhaps imagine them winding

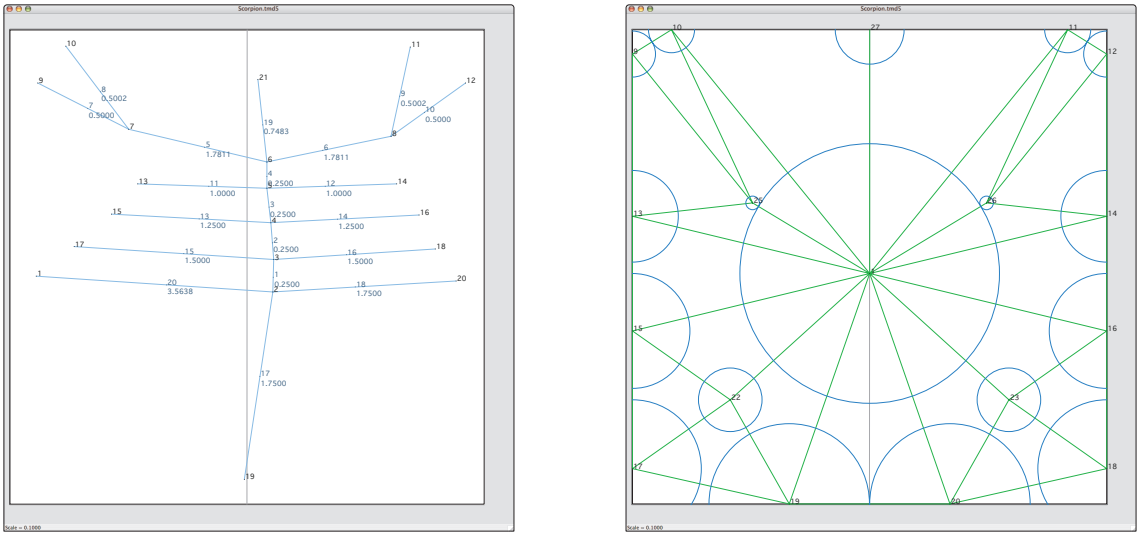

Figure 2: *Left: the stick figure as drawn within* TreeMaker. *Right: Result of the optimization (only circles are shown).*

between the circles.) Whenever two circles and the intervening rivers all touch one another, *TreeMaker* draws a line between the circle centers, shown here in green. These lines are actually some of the creases of the desired folded shape; they form a "skeleton" of the full crease pattern.

Conceptually, one finds the optimum packing by inflating all of the circles and rivers at the same rate with the circle centers trapped within the boundaries of the square and allowing the circles and rivers to move around during the inflation process until they are all wedged into position. This is, effectively, a nonlinear constrained optimization problem, and, stripped of the visual imagery of circles, rivers, and packing, it all boils down to a fairly straightforward algebraic description, which is what is needed for a computer implementation.

We assume that the tree graph can be described by a set of nodes $\{N_i\}$, edges $\{E_j\}$, and edge weights w_j, where the weight w_j gives the desired relative length of the jth flap of the desired base. For every path between two nodes $\{N_i, N_j\}$ we define the *path length* $l_{ij} \equiv \sum_k w_k$ as the (unique) sum of the weights of the edges between nodes N_i and N_j.

With every *leaf node* N_i of the tree graph, we associate a *vertex* $V_i = (x_i, y_i)$, which will be the center of the associated circle (and will also turn out to be the point in the square that maps to the tip of the corresponding flap). We further introduce the *scale factor* m, which sets the size of the folded shape relative to the original tree graph, so that if an edge of the tree graph had desired length w_j, its corresponding flap in the folded form will have actual length mw_j. The non-overlap condition can be expressed as an inequality:

$$|V_i - V_j| \geq m l_{ij}, \tag{1}$$

for every path l_{ij}. The requirement that circle centers lie within the square is similarly expressible as inequalities:

$$0 \leq x_i \leq 1, 0 \leq y_i \leq 1, \tag{2}$$

for every vertex V_i. For ease of foldability, one would like the folded form as large as possible relative to the size of the square; in other words, we should like m to be as large as possible. The solution of the origami optimization problem then becomes a case of maximizing the scale m subject to both sets of inequality constraints. And so, the solution of the arrangement of vertices requires the solution of a nonlinear constrained optimization, with linear and quadratic inequality constraints and a nonconvex feasible region.

A nice feature of this framework is that it is relatively easy to incorporate additional requirements into the basic optimization; for example, one can impose mirror symmetry on the crease pattern by adding additional equality constraints on the coordinates of the affected vertex pairs.

This type of constrained optimization is one that has been studied for decades within computer science. In my first few implementations of *TreeMaker*, I wrote my own constrained optimizer using the Augmented Lagrangian method [11]. On a 68K-based Macintosh (with no floating-point coprocessor), this took several minutes to converge to a solution, and so to make the wait more tolerable, I wrote code that updated the screen after each iteration, so I could watch the circles jostle around and expand, which offered entertainment value, if nothing else. For *TreeMaker 4*, I replaced it with a Feasible Sequential Quadratic Programming (FSQP) algorithm developed by Andre Tits at the University of Maryland [12], which turned out to be about a hundred times faster than my relatively crude code, but it also occasionally sent the evolving solution off into distant corners of feasible solution space—a behavior I was willing to tolerate, given the incredibly improved speed.

But there was another factor at work at the same time; throughout the 1990s, the most effective way to speed up code is often "wait for a faster processor." From the floating-point-in-software 68K Mac to current Intel processors, floating-point processing sped up by a factor of about a thousand, and so for *TreeMaker 5*, I went back to my hand-rolled ALM code, which was slow, but numerically efficient (and could be made open-source). On a modern PC, even this far-from-optimum code still converges almost instantaneously to a high-quality local optimum. Which is not necessarily the global optimum. Artistically, this is a good behavior: it lets the artist find several candidate arrangements by starting from different initial positions, and then choosing the local optimum that best fits other aesthetic criteria.

Whatever algorithm is used, at the end of the optimization, one has the positions of the vertices V_i and the scale factor m. That is sufficient to construct the circle centered on each vertex, as shown in Figure 2 (right). When two circles and all intervening rivers touch, it is an indication that the associated inequality constraint is, in fact, an equality. In the language of constrained optimization, that constraint is said to be *active*, and so we say that the path between the two nodes is *active* as well. It turns out that active paths always correspond to crease lines in the crease pattern associated with a circle packing. These creases turn out to lie along the axis of the folded shape, and so we also call them *axial paths*. They form the framework upon which the rest of the crease pattern is constructed.

3 The Creases

Once we have found the axial paths, it is possible to mathematically construct all of the other folds in the desired shape (which we call a "base")—again, a process that takes a fraction of a second in *TreeMaker*, this time because no optimization is needed; the construction is a straightforward polynomial-time geometric algorithm. *TreeMaker* color-codes the creases according to their structural role: there are axial creases (green), which all wind up along the centerline of the folded model, hinge creases (blue), that run along the junctions between flaps, and ridge creases (red), that define how wide each flap is. These are shown in Figure 3. The creases outline regions of the paper, called facets; the collection of all facets and their intervening creases gives the crease pattern needed to fold the square into the origami base.

The structural information indicated by color tells the designer where in the folded form a given crease lies and provides some hint for how the folded shape goes together, but ultimately, we need to determine whether each crease is a mountain or valley fold. To do this, *TreeMaker* computes the "stacking order" of all of the layers of paper, which is encoded by a graph on the facets that defines how the layers stack. Once we know the stacking order of the layers of paper, we can determine whether each fold is a mountain fold (solid black), valley fold (dashed colored), or is unfolded, or flat (solid gray). The full crease pattern, as computed

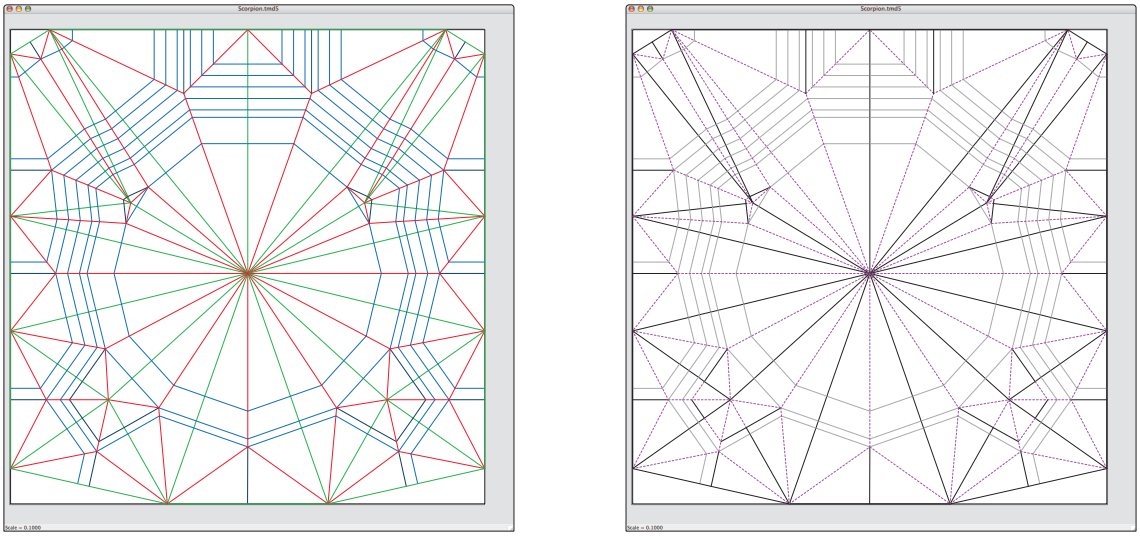

Figure 3 : *Left: The computed crease pattern with structural coloring. Right: Final crease pattern with mountain-valley assignment.*

by *TreeMaker*, is shown in Figure 3. Now, it's time to fold.

4 Folding

The design is only half the battle of creating an origami figure. Actually, it's even less than half. First, we have to fold the crease pattern into the base; then we have to turn the base into the finished form. All we are guaranteed from the algorithm is that the flaps will be of the right number, have the right length, and be connected to each other in the way we originally specified. But there are no guarantees on the flap width, cross section, arrangement of layers, or much of anything else. And at this point in the process, we don't even have a base: we just have the crease pattern for it.

And so, that's what we start with: folding the crease pattern into the base. To accomplish this, we must first transfer the crease pattern from the design to the actual square[1]. This is done by measuring the positions of a few key points, and then forming all of the other creases based on those initial measurements, as shown in Figure 4.

Most complex origami bases have the property that you can't fold them one crease at a time; instead, you must bring all of the creases together at once. That is the case here, so we bring all of the edges of the square together in a single motion. This process is called "the collapse" in origami, because we are "collapsing" the square down to the folded base, as shown in Figures 4–5.

"Collapse"-type folds had appeared in origami occasionally throughout the years, but its ubiquity in complex algorithmic designs came as something of a surprise in the origami world. Published origami designs almost always consisted of a sequence of relatively simple steps, leading smoothly from the initial square to the finished form with no individual step requiring more than three or four creases to happen at once. But complex modern designs are not like that; in fact, it is relatively rare that such a simple folding pattern exists for a complex crease-pattern-based design. Instead, they almost always require all creases to be brought together at once—a process my colleague Brian Chan has dubbed (somewhat mischeviously)

[1]The paper here is a 40 cm square of abaca/hemp paper made by Michael LaFosse, a master papermaker (`http://www.origamido.com`) and origami artist whose handmade papers are prized by origami insect folders around the world.

Figure 4 : *Left: Square with creases transferred to the paper. Right: First stage of the collapse.*

"three easy steps: precrease, collapse, shape." But in each case, the precreasing, the collapsing, and most especially, the shaping, may require several hours for each step.

In this figure, the collapse is relatively simple – only a few tens of creases are involved. (Truly advanced designs may have hundreds of creases participating in the single collapse step.) The result at the end of the collapse step is a flat form: the origami base, shown in Figure 5. It doesn't look like much—certainly, not like a scorpion—but it contains a flap of exactly the right length, connected in exactly the right way, for every leg, arm, pincer, and body segment of the desired subject.

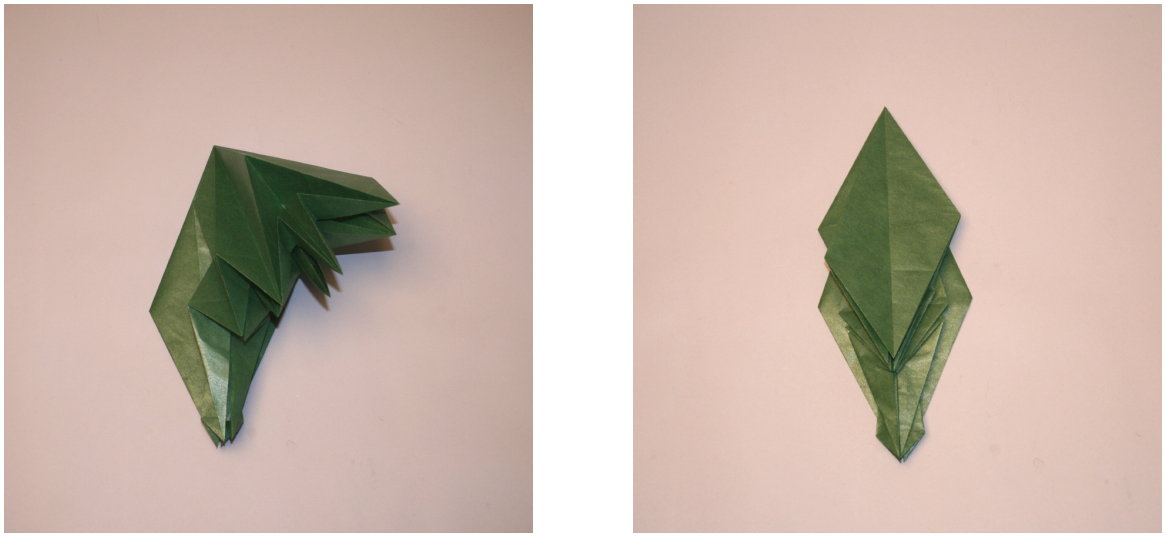

Figure 5 : *Left: The collapse under way. Right: The fully collapsed base.*

The flaps have the right length and connectivity, but not the right width; in fact, most of them are much wider than their corresponding body part. As problems go, that's a pretty good one to have; it's a lot easier to make a fat flap narrower than to make a narrow flap wider, and with arthropods as subject matter, making flaps narrower is the usual requirement. We will need to narrow all parts of this base so that they can better represent the skinny legs, arms, and tail of the scorpion. This is done with a series of what are called "sink

folds," which are, essentially nested sets of inversions of the layers of each flap. Sink folds are tedious to perform (any origami teaching session is guaranteed to elicit a series of groans when a set of sink folds is announced), but they are conceptually straightforward to perform. Figure 6 show the base with some (left) and all (right) of its layers sunken. In the end, the folded paper shape looks not unlike the original tree graph: a collection of long, skinny, connected sticks, each of which will become an appendage of the subject.

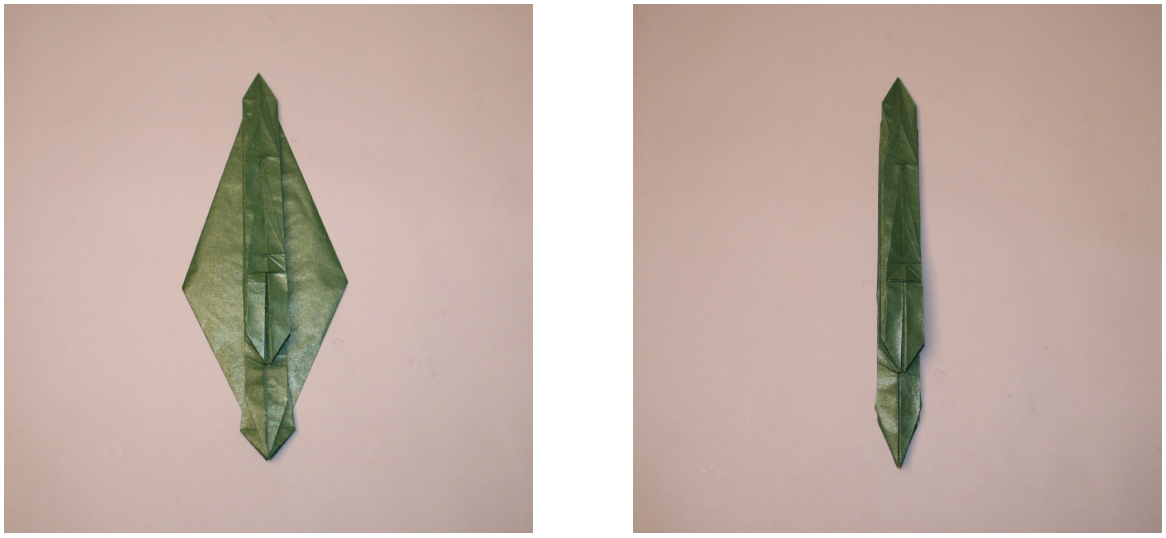

Figure 6 : *Left: The partially narrowed base. Right: The fully narrowed base.*

Last comes the shaping. There is no algorithm for this—and, in fact, this is one part of the process where both craftsmanship and artistry are preeminent in the process. A representional origami figure, no matter how realistic, is, in any absolute sense, still a rather crude approximation of the subject. The artist selects what features to emphasize and what features to minimize in order to convey the essential character of the subject. Craftsmanship is called for, too, to insure that every visible edge, every visible fold falls in the right place to convey the desired impression. A misplaced edge, an unintended wrinkle—these can spoil the visual effect, for the human eye is drawn to lines, ignores smooth surfaces. It is often the case that the shaping takes by far the greatest amount of time when creating a complex origami figure, and this process can last, literally, for days. (For this scorpion, shaping took only a few hours in total.) Figures 7–8 show the final process of shaping, beginning with narrowing of the legs, then proceeding through shaping of the tail and claws, and ending up with the final scorpion form. The narrowing and shaping of the legs is accomplished by selectively dampening flaps as they are folded, a technique called "wet-folding" that was introduced by the great 20th-century origami master Yoshizawa; strips of drafting tape hold the figure in position as it dries.

5 Final Thoughts

The early successes of circle-river packing and its relatives were most vividly seen in the subject matter of origami insects and their kin. In the 1970s, origami insects were considered the ultimate challenge, and at least one origami book flatly declared that an origami grasshopper was "impossible" from a single square. In the 1980s, grasshoppers were not only possible, but folded and then surpassed; and the 1990s saw a golden age of origami insects informally known as the "Bug Wars," in which origami creators vied with one another for the most realistic, leggiest, pointiest, wingiest, horniest creature they could find! (Entomology journals saw a noticeable uptick in their origami subscribers, as origami artists sought out the most challenging possible subjects to take on.) (See [2, 7] for some of my own contributions to this battle.)

Figure 7 : *Left: Narrowing the legs. Right: The curved tail has been folded and shaped.*

Figure 8 : *Left: Final stages of shaping. Right: The finished scorpion, approximately 15 cm long.*

To some degree, the Bug Wars are still going on, but with far more than bugs as their subject matter—complex origami subjects include other animals, plants, human figures, and indeed, purely geometric forms—and the goals of origami artists have moved well beyond "mere realism," to now address the question,"how can I create a visually interesting shape that challenges the mind and perceptions of the viewer?" Design techniques have grown as well; circle-river packing has been expanded to include techniques based on packings of polygons and other shapes, and the ranks of other algorithms for computational origami continues to grow, with contributions now coming from the academic world of computational geometry and other parts of mathematics and computer science. It is a heady time.

And yet, all of these techniques are still no more than tools in the hands of the artist. I am sometimes asked if the development of mathematical design techniques (or, heaven forbid, *computer programs*!) somehow diminishes origami as an artistic activity. To which I answer: these techniques, for all their power, are still just tools, no more than a painter's pigments, or a sculptor's chisels. The quality of the tool does not set the quality of the artwork; what matters most is what you do with those tools. But seeing what origami artists from all over the world have done in recent decades with the tools that computational origamists have given them, and an entirely new generation of conceptual and computational tools coming down the pike, I have no doubt that the future of origami as an art form will be exciting and invigorating for many years to come.

TreeMaker is open-source (GPL), runs on Macintosh OS X, Linux, and Windows computers, and may be downloaded at [8].

References

[1] Robert J. Lang. Mathematical algorithms for origami design. *Symmetry: Culture and Science*, 5(2):115–152, 1994.

[2] Robert J. Lang. *Origami Insects and their Kin*. Dover Publications, 1995.

[3] Robert J. Lang. A computational algorithm for origami design. In *12th ACM Symposium on Computational Geometry*, pages 98–105, 1996.

[4] Robert J. Lang. The tree method of origami design. In Koryo Miura, editor, *Origami Science and Art: Proceedings of the Second International Meeting of Origami Science and Scientific Origami*, pages 73–82, Ohtsu, Japan, 1997.

[5] Robert J. Lang. Scorpion, opus 379, 2002. `http://www.langorigami.com/art/gallery/gallery.php4?name=scorpion_varileg`.

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

[7] Robert J. Lang. *Origami Insects II*. Gallery Origami House, 2003.

[8] Robert J. Lang. TreeMaker, 2003. `http://www.langorigami.com/treemaker.htm`.

[9] Toshiyuki Meguro. Jitsuyou origami sekkeihou [practical methods of origami designs]. *Origami Tanteidan Shinbun*, 2(7–14), 1991–1992.

[10] Toshiyuki Meguro. 'Tobu Kuwagatamushi'-to Ryoikienbunshiho ['Flying Stag Beetle' and the circular area molecule method]. In *Oru*, pages 92–95. 1994.

[11] R. T. Rockafellar. Augmented Lagrangian multiplier functions and duality in nonconvex programming. *SIAM Journal on Control*, 12(2):268–285, 1974.

[12] Andre Tits. CFSQP. `http://www.aemdesign.com`.