

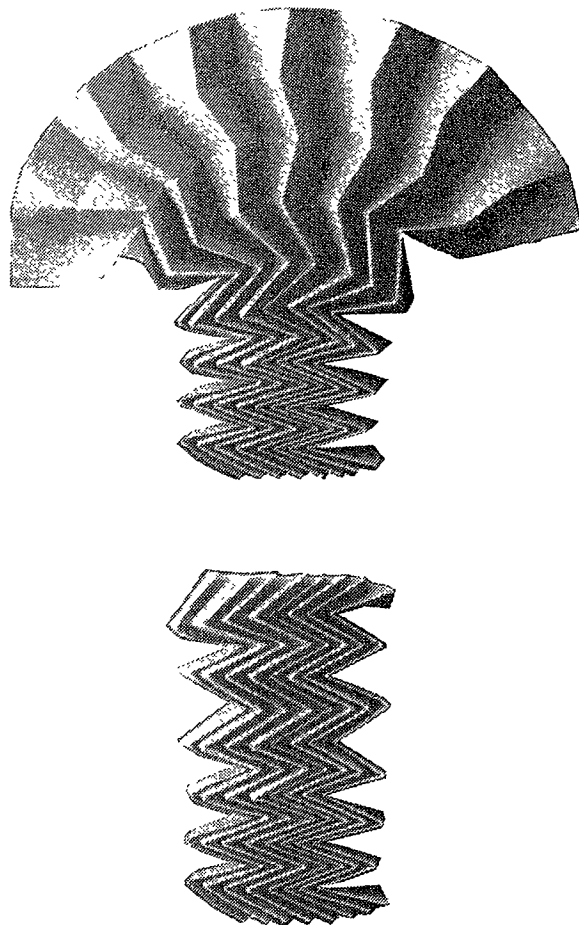
Symmetry: Culture and Science

Origami, 2

The Quarterly of the
International Society for the
Interdisciplinary Study of Symmetry
(ISIS-Symmetry)

Editors:
György Darvas and Dénes Nagy

Volume 5, Number 2, 1994



The *Miura-ori*
opened out like a fan

MATHEMATICAL ALGORITHMS FOR ORIGAMI DESIGN

Robert J. Lang

7580 Olive Drive Pleasanton, CA 94588, USA
E-mail: rjlang@aol.com

Although hundreds of years old, the Japanese art of origami has only recently become the subject of mathematical scrutiny. In recent years, a number of mathematical aspects of origami have been published in books and journals. A sampling of the work of mathematical folders is to be found in recent mainstream publications, e.g., (Kasahara, 1988) and (Engel, 1989); however a large number of folders have attacked the problem of systematic/mathematical origami design. They include Peter Engel and myself in America, and many folders in Japan, including Husimi, Meguro, Maekawa, and Kawahata. As befits a young and expanding field, much of the scientific analysis is circulated informally (notably over the origami-l mailing list on the Internet: to join, send the message "subscribe origami-l yourname" to listserv@nsts.nsl.ca).

The goal of many origami aficionados is to design new origami figures and for many, the pursuit of origami mathematics is a search for tools leading to ever more complex or sophisticated designs. In this article, I will describe two powerful algorithms for origami design that I have successfully applied to the design of fish, crustacea, insects, and numerous other origami models. Although I will describe the algorithms in the form I am familiar with, similar techniques have been described by Dr. Toshiyuki Meguro in the (Japanese-language) publication *Oru* and in the newsletter of the *Origami Tanteidan*, a Japanese association of origami designers.

1. THE CIRCLE METHOD OF DESIGN

The first design approach is represented by what I and others call the 'circle method'. In the circle method, each flap on the origami model is represented by a circle whose radius is equal to the length of the flap. The goal of the origami design process is to place circles representing each flap on the square in such a way that the centers of all circles lie within the square (although some part of the circle can extend over the edges of the square) and no two circles overlap one another. This

approach is one that both I and Fumiaki Kawahata have used extensively, although – as happens so often in the sciences – we each developed our methods initially unaware of the other’s activities. I am not aware of other Westerners using the circle method, although the young American folder, Jimmy Schaefer, has developed a successful and related design method, which he has dubbed, ‘the method of isolating squares’, based on concepts similar to the circle method. In Japan, these concepts of origami design are more widely known than in the West.

The fundamental concepts of the circle method of design and its derivatives are two: first, since most origami models can be broken down into a number of flaps of various lengths, a successful design hinges upon constructing the right number and sizes of flaps. Second, paper must be conserved; any part of the square can be used in no more than 1 flap at a time. The circle method of origami design consists of representing the subject as a collection of flaps and allocating a unique circular region of paper for each flap.

For the purposes of origami design, there are three different types of flaps: ‘corner’ flaps, ‘edge’ flaps, and/or ‘interior’ flaps, or ‘middle’ flaps, as some call them. The different types of flaps are named for the point where the tip of the flap falls on the square. If you take a model that has a lot of flaps, color the tip of each flap, and then unfold the model, you’ll get a square with a pattern of dots on it. Some dots will fall on the corners of the square; others on the edges; still others will be in the interior of the square. Since each dot corresponds to a flap of the model, we can classify the flap by the location of the dot, which is the location of the tip of the flap. A corner flap has its tip come from a corner of the square, an edge flap has its tip lie somewhere along an edge, and a middle flap, as you would expect, comes from the middle of the paper. For example, the four large flaps on a Frog Base are corner flaps; the four stubby flaps are edge flaps; and the thick flap at the top is a middle flap.

The reason for the distinction between the three different types of flap is that for a given length, each of the three types of flaps consumes a different amount of paper. One way to see this difference is to fold corner, edge, and interior flaps of exactly the same size from three different squares as shown in Figure 1 below, where I illustrated the folding of a corner flap. If you imagine (or fold) a boundary across the base of the flap, then that boundary divides the paper into two regions: the paper above the boundary is part of the flap, and the paper below the boundary is everything else. The paper that goes into the flap is for all intents and purposes consumed by the flap; any other flaps must come from the rest of the square.

So, as Figure 1(a) shows, if you fold a flap of length L from a square so that the tip of the flap comes from the corner of the square, when you unfold the paper to the original square, you see that the region of the square that went into the flap is roughly a quarter of a circle. Well, technically, it’s a quarter of an octagon. Suppose we made the flap half the width, as shown in Figure 1(b) before we unfolded it;

then the flap becomes a quarter of a 16-gon. If we kept making the flap thinner and thinner (using infinitely thin paper!), the boundary of the flap would approach a quarter-circle. Since we may not know ahead of time how thin a given flap will be, we'll take the circle as a reasonable approximation of the boundary of the region of the paper consumed by the flap. A corner flap of length L , therefore, requires a quarter-circle of paper, and the radius of the circle is L , the length of the flap.

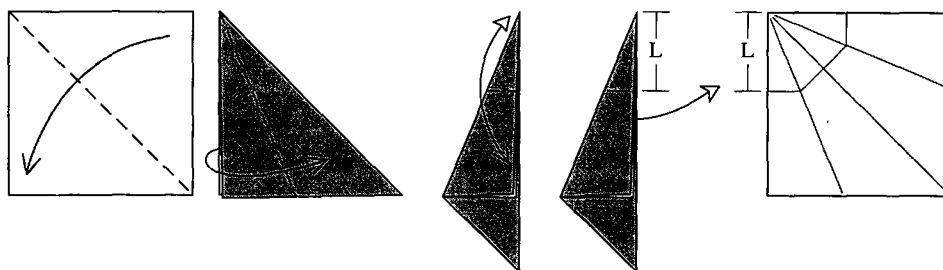


Figure 1(a): Folding a corner flap of length L from a square.

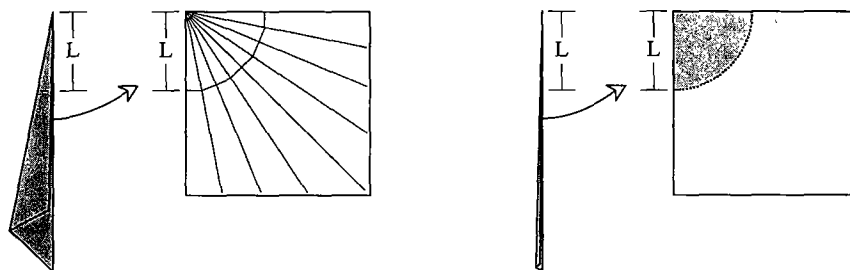


Figure 1(b): (Left) Making a narrower flap makes the boundary a quarter of a 16-gon. (Right) The limit of the boundary as the flap becomes infinitely thin approaches a semicircle.

Therefore, all of the paper that lies within the quarter-circle is consumed by the flap, and the paper remaining is ours to use to fold the rest of the model.

Now, suppose we are making a flap from an edge. How do we do that? Well, if we fold the square in half, then the point where the crease hits the edge become corners, and we can fold corner flaps out of one of these new corners, as shown in Figure 2. If we fold and unfold across the flap to define a flap of length L and then unfold to the square, you see that an edge flap of length L consumes a half-circle of paper, and again, the radius of the circle is L , the length of the flap.

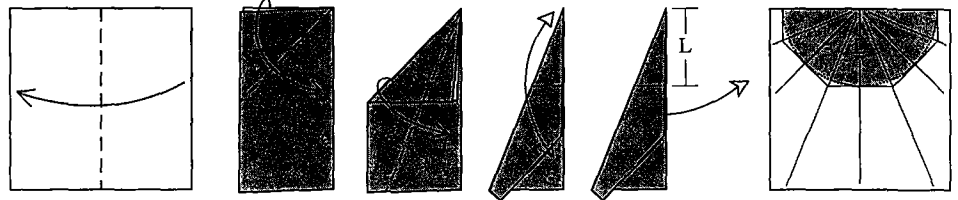


Figure 2: Folding an edge flap of length L from a square.

Similarly, we can make a flap from some region in the interior of the paper (it doesn't have to be the very middle, of course). Figure 3 shows how such a flap is made. When you unfold the paper, you see that an interior flap requires a full circle of paper, and once again, the radius of the circle is the length of the flap.

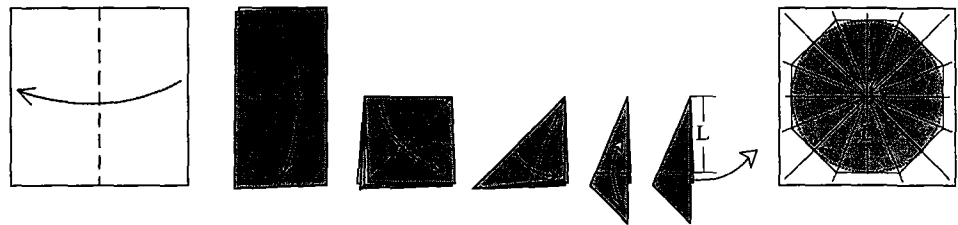


Figure 3: Folding a middle flap of length L from a square.

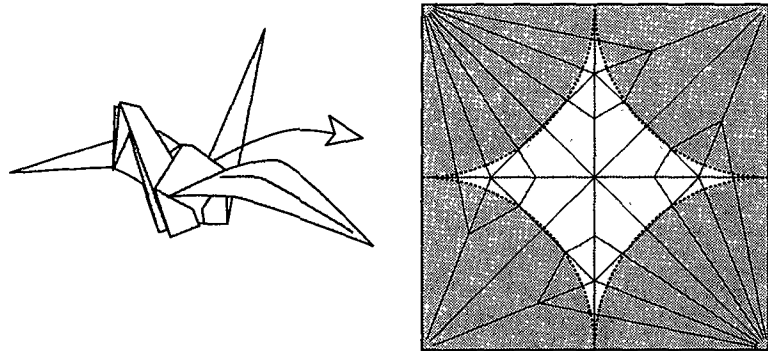


Figure 4: Unfolding a crane (which has four major flaps) reveals that the quarter-circles corresponding to the four flaps consume almost all of the paper in the model.

So, any given flap in a model consumes a quarter, half, or full circle of paper, depending upon whether it is a corner flap, edge flap, or interior flap. It is an interesting and illuminating exercise to unfold an existing model and draw in the

circles corresponding to the various flaps of the model. For example, the traditional crane, made from the Bird Base, has four major flaps. As shown in Figure 4, they are all corner flaps, and each flap consumes a quarter-circle of the square. (If we count the pyramid in the middle of the back as a flap, we would have another, smaller circle in the center of the square. However, since real cranes don't have pyramids in their back, I consider that an 'accidental' flap and we won't count it in our tally.) Since well over $3/4$ of the area of the square goes into the four major flaps, we would say that the crane shows an efficient use of paper.

What we are doing here is to build up a set of mathematical tools that can be used to design origami models, making origami design a scientific process. One of the goals of all scientific endeavors is the concept of unification: describing several disparate phenomena as different aspects of a single concept. Rather than thinking in terms of quarter-circles, half-circles, and full-circles for different kinds of flaps, we can unify our description of these different types of flaps by realizing that the quarter-circles, half-circles, and full circles are all formed by the overlap of a full circle with the square, as shown in Figure 5. The concept common to all three types of flaps is that the paper for each can be represented by a circle with the center of the circle lying somewhere within the square. With middle flaps, the circle lies wholly within the square. However, with corner and edge flaps, part of the circle laps over the edge of the square. (The center of the circle still has to lie within the square, though.) Thus, *any* type of flap can be represented by a circle whose center, which corresponds to the tip of the flap, lies somewhere within the square.

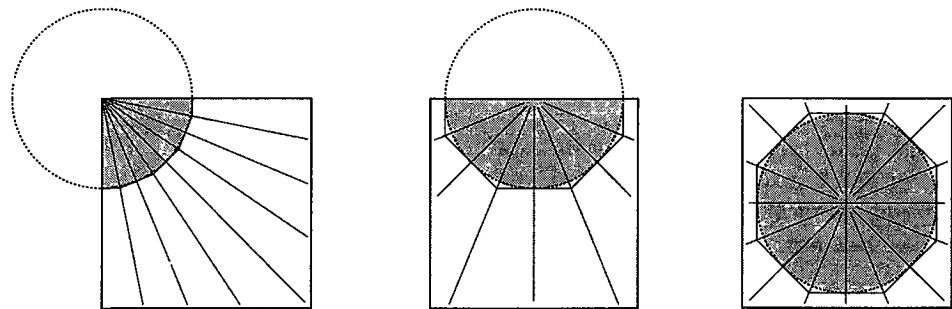


Figure 5: All three types of points can be represented by a circle if we allow the circle to overlap the edges of the square.

The examples above showed how to fold a single flap from a square. Suppose we want to make more than one flap at a time from the square (which is usually the case, unless you are designing a worm), and you draw the circles corresponding to each flap. Is there anything we can say about the circles even before we start? Well, since no part of the paper can be used in two different flaps simultaneously, and each circle delineates the paper used in each flap, no two parts of the paper can lie

inside two different circles. Therefore, *no two circles corresponding to different flaps can overlap on the unfolded square.*

Although this property seems pretty general, it is in fact quite restrictive. If you want to fold a model with ten flaps, you know that if you unfold the model and draw the circles corresponding to each flap, no two of the circles will overlap. So if you eliminate all arrangements of points for which the circles overlap, you must be closer to a design solution.

In fact, the ramifications of the non-overlapping property are a great deal stronger; if you draw ten non-overlapping circles on a square, it is *guaranteed* (in a mathematical sense) that the square can be folded into a base with ten flaps whose tips come from the centers of the circles. So merely by shuffling circles around on a square, you can construct an arrangement of points that can be folded into a base with the same number of points, no matter how complex!

Therefore, here is an algorithm for origami design:

- (1) Count up the number of appendages in the subject and note their lengths.
- (2) Represent each flap of the desired base by a circle whose radius is the length of the flap.
- (3) Position the circles on a square such that no two overlap and the center of each circle lies within the square.
- (4) Connect adjacent centers to one another with crease lines.

The resulting pattern can be foldable into a base with the number and dimension of flaps that you started with.

This is a powerful property for origami designers. If you lay out circles corresponding to the flaps of your subject on a square so they don't overlap, *you are guaranteed of the existence of a folding sequence that can transform the pattern into the desired base.* Finding the folding method may still be a bit of a trick, of course, but by beginning from a valid circle pattern, you certainly eliminate a lot of blind alleys.

One thing that is immediately apparent from the circle method of design is that corner flaps consume less paper than edge flaps, which consume less paper than interior flaps. Turn this property around, and you find that for a given size square, you can fold a larger model (with fewer layers of paper) if you use corner flaps rather than edge flaps, and edges flaps rather than interior flaps. Seen in the light of the circle method, the traditional crane – and the Bird Base from which it comes – is an extremely efficient design, since all four flaps are corner flaps, and almost all of the paper goes into one of the four flaps. However, add one or two more

flaps, and you are forced to use edge flaps. Once you start mixing edge flaps and interior flaps, you begin to run into tradeoffs in efficiency. Sometimes, it is even better *not* to use the corners for flaps if there are additional flaps to be placed on the square!

So, for example, suppose we want to fold a base with *five*, rather than four, equal-length flaps. A little doodling with a pencil and paper (or alternatively, you can cut out some cardboard circles and shuffle them around) will reveal two particularly efficient arrangements of circles, as shown in Figure 6 below.

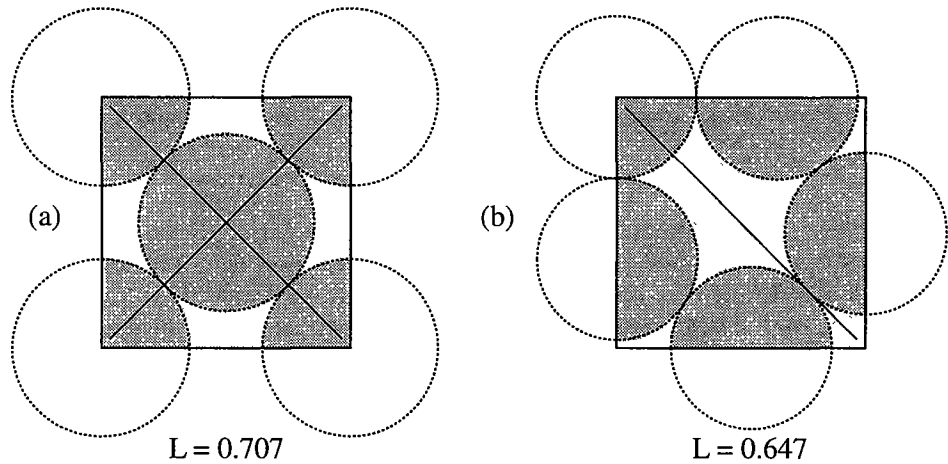


Figure 6: Two circle patterns corresponding to bases with five equal-length points.

Now we have two possible circle patterns. Which one is better? Is there any way to quantify the ‘quality’ of a crease pattern?

One way of comparing different ways of folding the same base is to compare their efficiency; that is, from a given size square, how large is the base? A useful measure of efficiency is to compare the size of some standard feature of the base – such as the length of a flap – to the size of the original square. To facilitate this comparison, let’s assume our square is one ‘unit’ on a side. If you’re using standard origami paper, a unit is 10 inches. For the crease patterns shown in Figure 6(a), if all of the circles are the same size, it is fairly easy to work out that the radius of each circle, and thus the length of each of the five flaps, is $1/\sqrt{2}$, or 0.707. For the pattern in Figure 6(b), it is somewhat harder to calculate but the radius of each circle is 0.647, or about 10% smaller. Thus, a five-flap base made from pattern 6(a) will be slightly larger, and slightly more efficient than the pattern made from Figure 6(b).

These two circle patterns are relatively simple. By connecting the centers of the circles with creases and adding a few more creases, you can collapse the model into

a base that has the desired number of flaps. As it turns out, there already exists in the origami literature two bases that correspond to these circle patterns, shown in Figure 7. Figure 7(a) is the circle pattern for the Frog Base, while Figure 7(b) is the circle pattern for John Montroll's 'Five-Sided Square' (Montroll, 1985).

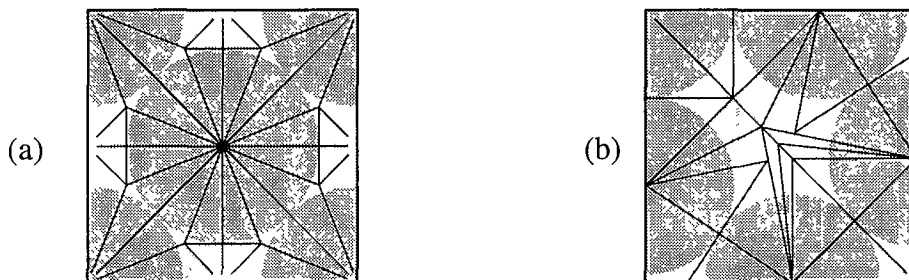


Figure 7: Full crease patterns corresponding to the two circle patterns.

You can also see a difference between the two bases. In the Frog Base, the fifth flap is a thick middle flap and points in the opposite direction from the four corner flaps; whereas in the Five-Sided Square, the four edge flaps and the corner flap go in the same direction and can easily be made to appear identical (which, of course, was the original rationale for John's design). It's worth a slight reduction in size to obtain the similarity in appearance for all five flaps.

While the two solutions for five equal-sized flaps correspond to published bases, I find it remarkable that the most efficient base for six equal flaps is not yet published. You might wish to try your hand at the following two problems:

- (1) Find a circle pattern for the largest possible base that has six equal-length flaps and fold it into the base.
- (2) Find a circle pattern for the largest possible base that has six equal-length flaps, 3 on each side of a line of bilateral symmetry, and fold it into the base. (The surprising solution has two middle flaps!)

As I said before, although the circle method guarantees that a folding sequence exists to convert the skeletal crease pattern into a base, it doesn't necessarily provide any guidance as to what that folding sequence actually is! (Meguro's 'molecular' approach of fitting together pre-existing crease patterns, however, helps fill in this gap.) So even if you work out a circle pattern, with or without a computer, you (and the computer) still have some work ahead of you to figure out how to fold the crease pattern into a base. However, it is a big help to start with a crease pattern that is guaranteed to work – you can avoid using a basic symmetry that is doomed to failure from the start!

Interestingly, there is a strong connection here between origami design and a well-known branch of geometry, that of packing circles. (For an excellent introduction to the latter, see (Gardner, 1992), Chapter 10, Tangent Circles.) For every pattern of circle packings in a square there is a corresponding origami base and vice-versa; conversely, many origami design problems may be solved by published solutions to different circle-packing problems.)

The circle method as described above works very well for models with many flaps that all come from the same part of the subject's body. The legs of insects and spiders, for example, all emanate from a single body segment. However, the circle method has one enormous liability. Since we consider *only* the total number of flaps and their lengths in the circle method of design, we have no way of incorporating information about how those flaps are connected to one another into the design. In fact, the circle method implicitly assumes that all the flaps are connected to each other at a single point! In the subject, the head bone may be connected to the neck bone, and the neck bone's connected to the chest bone, but with the circle method, every bone's connected to every other bone at one spot.

That is a severe limitation for origami design. For example, a typical mammal has three flaps (head and forelegs) at one end, three (tail and hind legs) at the other, and a body in between. The circle method can produce the head, legs, and tail, but there is no mechanism to include extra paper between the forelegs and hind legs to form a body. The circle method is not the whole story of origami design, however. There is a more sophisticated algorithm that includes the body, and in fact does indeed work for arbitrary arrangements of flaps and their connections. Now that we have established some basic concepts of origami design, we are ready to move on to the next level of origami design and introduce the 'tree method'. This new algorithm will be described in the next section.

2 TREE METHOD OF DESIGN

The circle method as described above works best for models that have all of their flaps emanating from nearly the same place, models whose basic shape is star-like. Quite a few subjects fall into this category – particularly insects, which have legs and wings all emanating from a single body segment, the thorax. (Antenna cause problems, since they come from the head.) However, the circle method gives less-than-satisfactory results for subjects that don't have a simple star shape – like most terrestrial vertebrates. A typical mammal, for example, has a cluster of three flaps (head, forelegs) separated from another cluster of three flaps (tail, hind legs) by an additional segment (the body). The circle method only deals in flaps and clusters of flaps; we have no way of including segments that connect different clusters together.

An extension of the circle method works for a much larger class of models that can have more complex structure. I call the extended method the ‘tree method’. Although the tree method is built upon the ideas of the circle method, it takes a somewhat different form. The fundamental concept of the tree method of origami design is that you represent the model by a stick figure (the tree) that has a branch for each arm, leg, wing, or other appendage. Each branch has a certain length, which you have chosen to be the length of the appendage in the final model. By mapping the tree onto a square according to a small number of rules, you can construct the skeleton of a crease pattern that, like the one you get from the circle method, is guaranteed to be capable of being folded into the stick figure and, by extension, into the desired model. Unlike the circle method, which only applied to stick figures that were fundamentally star-like, the tree method works for arbitrarily connected graphs.

To understand the rules for the mapping, I’ll show how it applies to a realistic design problem, a lizard. A lizard is simple enough that we won’t get bogged down in a lot of details, but it’s complicated enough to illustrate the basic approach, and because of its long body, it is precisely the type of model that the circle method has trouble with.

So, Figure 8(a) shows a drawing of a lizard. We would like to fold an origami model of a lizard, that might look something like Figure 8(b). (Actually, I’d hope it looks a lot better than Figure 8(b); I can fold a lot better than I can draw.) I’ll start by drawing the ‘tree’ that represents the lizard, as shown in Figure 8(c). A tree is a stick figure. On this stick figure, I have labeled each branch of the tree – which corresponds to an appendage or body segment of the lizard – with its desired length. The tail and body are 2 units long, the legs are 1 unit long each, and the head is also 1 unit long. Since I don’t know what size square I’m going to be folding from and I don’t know (yet) how large the lizard is with respect to the square, I’ll defer, for the moment, the question of just how long a ‘unit’ is.

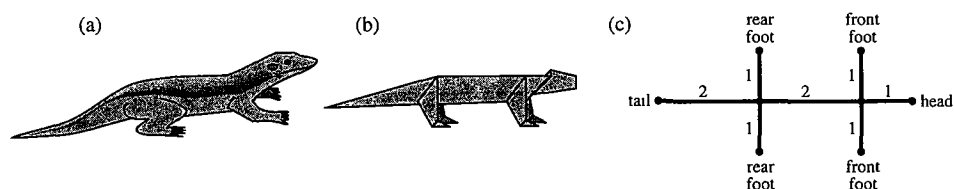


Figure 8: (Left) A real lizard. (Middle) A hypothetical origami lizard. (Right) The tree, or stick figure, corresponding to our hypothetical lizard.

My end goal is to find a crease pattern on a square that can be folded into the lizard. My intermediate goal will be to find a crease pattern that can be folded into

the tree. At first, it seems like I've made my life harder by setting the tree as my target. Since the branches of the tree are infinitely thin, it would take an infinite amount of folding (and infinitely thin paper) to fold it exactly. However, think of the tree as the lizard stripped of confusing detail. The tree, in its stark simplicity, represents an easier target for origami design than the original subject. And anyhow, I don't have to fold the tree exactly; if I fold a shape that closely resembles the tree, I'll have a shape – a base – that is suitable for folding a lizard.

In the tree method of design, each branch on the tree corresponds to some flap on the square. Tree branches that end at a point – which I call 'leaves' (or terminal nodes, if you prefer) – correspond to appendages of the subject, like the head, tail and legs. Tree branches that are connected to other branches at both ends correspond to body segments that join groups of appendages.

Since I'm trying to establish a link between the square and the hypothetical lizard, let's start at the lizard and work backwards. If I already possessed a folded version of the lizard and I made dots at important points – the tips of the legs, head, and tail, and where legs and body come together – when I unfolded the paper to a square, I could keep track of where those significant points fall on the square. The points where branches terminate or where several branches come together are important; I'll call each of those points a 'node' and label it with a name, corresponding to its position in the subject. Obviously, all of the nodes must lie *somewhere* on the square, and a great deal of the structure of the base is tied up in where the different nodes fall on the square. We ought to be able to draw the entire tree on the square so that the nodes match up with their corresponding points. There are lots of different possibilities for the position of the nodes with respect to the square; a few of them are shown in Figure 9.

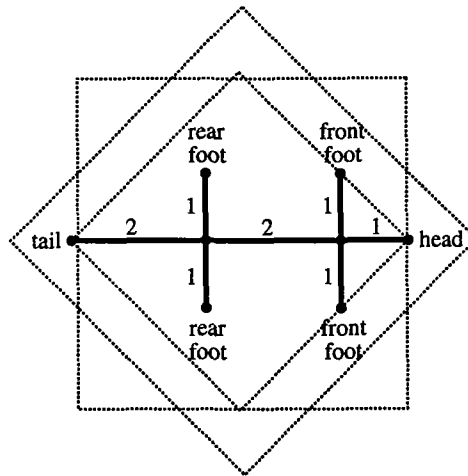


Figure 9: Three different possible arrangements of the lizard tree on the square.

It sort of makes sense that we would want the head and tail at opposite corners of the square, so let's suppose for the sake of argument, that we already had a successfully folded lizard-like shape, that we marked the locations of the nodes and branches on the shape, and then unfolded it to a square, giving the arrangement shown in Figure 10.

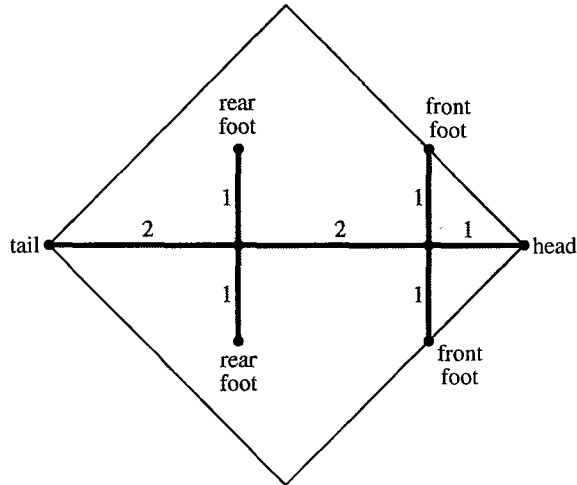


Figure 10: One possible arrangement of nodes for the lizard tree.

An important issue (actually, one of the most important) is how large the tree is compared to the size of the square. One of the hallmarks of good origami design is efficiency; the best designs generally are those that waste very little paper. For example, if you are folding a 3-pointed shape, you could start from a 4-pointed base and crumple up one of the flaps to hide it; but that wouldn't be nearly as esthetically pleasing as to work from a 3-pointed base from the very beginning. It's poor form to have to hide an unwanted flap, or even to have to make a flap drastically shorter by folding it in half. In our design, we don't want to have points or flaps that serve no purpose. So, in our design, we should use as much of the paper as possible for the designed parts of the figure and have no extra paper left over.

Also, the most efficient designs generally have the fewest layers, at least, when compared to designs of comparable complexity. Generally, the smaller a shape is, the more layers it has. That means that for a given size square, the base that we design should be as large as possible; consequently, the size of a unit of length of the tree should be maximized.

Let's figure out how big the tree is for Figure 10. The distance from the head to the tail is 5 units (2 for the tail, 2 for the body, and 1 for the head). The diagonal of a square is about 1.4 times the side of the square, and as I've shown, the diagonal is equal to 5 tree units; thus one tree unit is $1.4/5 = 0.283$ times the side of the square. This quantity – the ratio between a tree unit and the side of the square – is an important measure of the efficiency of a design, and we'll call this the 'scale' of the crease pattern. The larger the scale is, for a given size square, we get a larger base with fewer layers, a base that is more efficient and (one hopes) more esthetically pleasing.

The main problem with the arrangement of nodes depicted in Figure 10 is that it doesn't work. If you try to fold a lizard using this arrangement, you will find that, although the body and tail are easy to make, the legs come out rather shorter than we intended. In fact, the back legs will be almost nonexistent. Furthermore, quite a lot of paper at the top and bottom corners goes essentially unused. It doesn't seem right that the base comes out with the wrong proportions and there is unused paper as well! So we can't just draw the tree to scale on the square and expect things to work out. Obviously, we're overlooking some crucial concept. There must be some additional rule to be applied that limits the size of the tree when it is mapped onto the square.

Well, of course with all the folding that goes on between the square and the base, the tree pattern on the square could get rather distorted. We can establish some limits on the amount of distortion, though. Consider the following thought experiment. Suppose an ant wishes to walk from the tail of the lizard to one of the rear feet. On the lizard, she starts at the tip of the tail, walks up the tail to the flap where the tail and legs meet, turns, and walks down the leg. The distance the ant has walked is the length of the tail plus the length of the leg, or, as I've drawn in Figure 11, a total of 3 units.

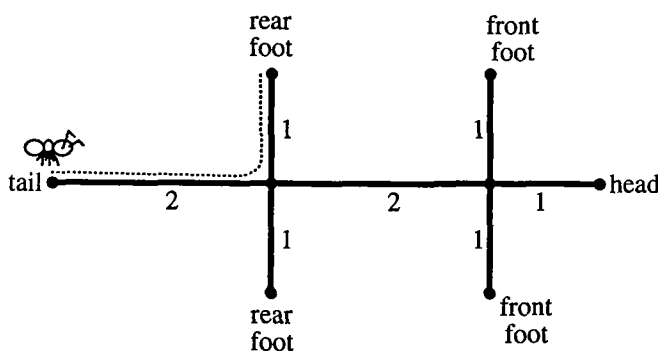


Figure 11: Picture an ant walking from the tail to the rear leg. She can't go by the shortest route (as the crow flies); instead, she must walk up the tail and back down the leg, for a total of three units.

Suppose that just before the ant set out, we dipped her in ink, so that when she walked, she left a trail of ink soaking through the paper. Now we unfold the base and look at the various trails left by the ant. Since in most origami bases, each flap consists of several layers of paper, the ant will probably have left several trails between the two nodes. Depending on the folding pattern for the base, some of the trails might weave around a bit, while others go more directly from the tail to the foot. Several possible trails are shown in Figure 12.

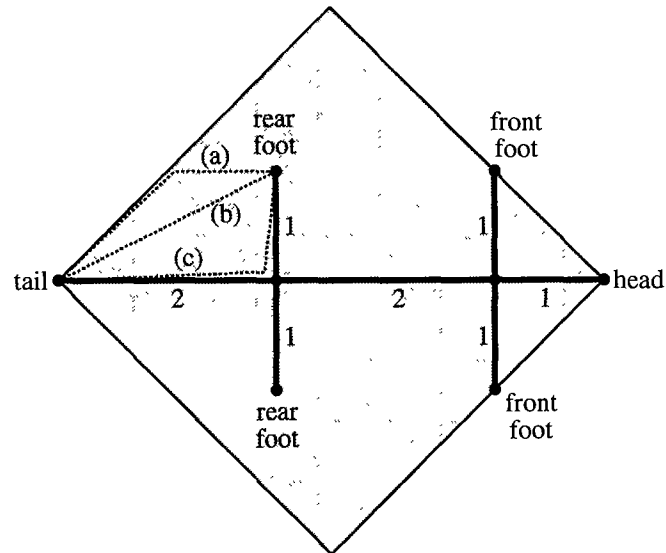


Figure 12: Three paths from tail to rear foot.

Although we may not know what ground was covered by the ant, we do know that the ant walked exactly 3 units on the tree. If in the folded base some paper was doubled back on itself, then some of the paths on the square might be longer than 3 units, but no path can be *shorter* than 3 units. In particular, the *shortest* possible ink trail is the one that runs directly between the tail node and the leg node (trail (b) in Figure 5), and it, too, must be *at least* 3 units long. Thus, we know that a successful crease pattern must have the tail node and each leg node separated by a minimum of 3 units.

A similar condition exists for every possible pair of nodes. If the ant goes from the tip of the tail to the tip of the head, he travels $2+2+1=5$ units; thus the tip of the tail must be separated from the head on the square by 5 units. From one front leg to the other is $1+1=2$ units, so the two front leg nodes must be separated by 2 units; and so on and so forth.

This condition must hold for any pair of nodes: the distance between two nodes on the square must be *at least* the distance between the two nodes *measured along the*

branches of the tree. Nodes connected by a single branch must be separated by at least the length of the branch; nodes connected by two branches must be separated by the sum of the lengths of the two branches; and so on, for every possible pair of nodes.

Now you see the problem with the tree structure of Figure 10. The way I've drawn it, although all nodes connected by a single branch are separated by the proper amount, the shortest path between the nodes corresponding to the tail and rear legs is only $\sqrt{3}$, or about 2.2 units long, when it should in fact be at least 3 units.

The same type of shortfall affects the paths between the front and rear legs. As I've drawn them in Figure 10, the front and rear feet on one side of the body are separated from each other by the body length, or 2 units; but in fact, they need to be separated from each other by 4 units, since our proverbial ant must walk down one leg, along the body, and back out the other leg to go from toe to toe. Similar shortfalls afflict the paths between the head and front legs, between the head and rear legs, between the tail and front legs, and between the front and back legs. Figure 13 shows all possible paths drawn in on the tree and all of their minimum lengths.

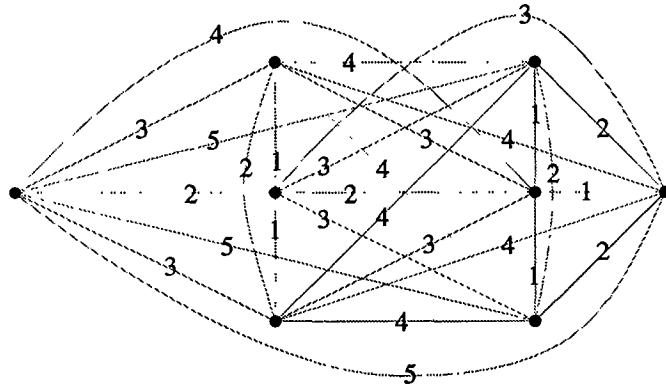


Figure 13: Paths and lengths for all nodes on the lizard tree.

In general, many pairs of nodes on the square must be farther apart from one another than they are 'as the crow flies' on the tree. There are two ways we could overcome this problem. We could keep the arrangement of nodes as we have them in Figure 11 and multiply all distances by some fixed value. A quick check of all possible paths shows that the paths in Figure 10 in the worse shape are the ones between front and rear legs on one side, which must be separated by 4 units. If we simply double all distances so that the tail node is separated from the hip node by 4 units, the leg nodes are separated from the hip nodes by 2 units and so forth, then the leg nodes are separated from one another by the required 4 units, and in fact all pairs of nodes meet their minimum separation requirement.

But doing this means making our tree unit smaller. After scaling down the tree by a factor of 2, the diagonal of the square is 10 tree units long, so that the scale of the crease pattern has fallen from 0.28 to 0.14.

But this is awfully wasteful! Although now the path between front and rear legs are equal to their minimum length, *all* the other paths are longer than they have to be, which means that we'll be wadding up excess paper to get each point down to its proper length. What we really ought to do is to increase the size of a unit *and* rearrange the nodes to allow a larger unit size.

Since the lizard is a pretty simple shape, it's easy to see that we can improve the design by moving the nodes corresponding to the feet out to the edges of the square and, since the tail is longer than the head, moving the rear feet farther from the tail corner than the front feet are from the head. Figure 14 shows an optimum distribution of nodes for the lizard. Even for the optimum, most of the paths turn out to be longer than their allowed minima. (In fact, you have considerably freedom in your placement of the nodes corresponding to the hips and shoulders.) The paths that turn out to be the limiting paths – which I call the 'critical paths' – are the ones from head to forelegs, from forelegs to rear legs, and from rear legs to tail. I've made the critical paths heavy in Figure 7. If you work out the geometry (or just draw it to scale and measure), you find that one tree unit is $(\sqrt{31}-5)/3$ times the side of the square, which works out to a scale of 0.189 – smaller than what you get from Figure 10, but now it's foldable. Since we know the scale, we can also figure out how big the final model will be: the base we fold from this pattern will make the length of each leg about one-fifth the side of the square, and from nose to tail, the model will be almost as long as the side of the square.

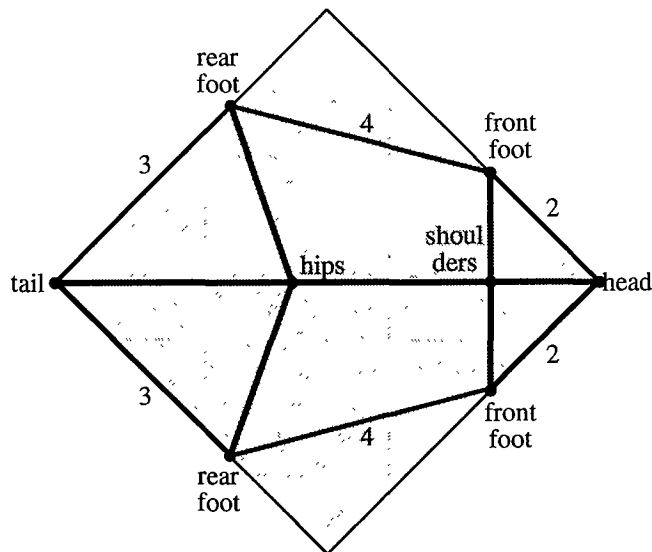


Figure 14: An optimum distribution of nodes on the square for the lizard tree.

Now, we have a crease pattern that satisfies all of the criteria we have set so far. Can it really be folded into a base of the proportions we have set? The answer is yes, it can. Figure 15 shows all the creases for one of many possible ways to fold this into a base. While the base itself is probably what you would fold a lizard from, by repeated box-pleated sinks, you can transform the base into a pretty fair approximation of the stick figure, the tree, itself. Of course, you don't need to go that far. The useful result of this little exercise is the second figure in Figure 15; the base, which we construct as a byproduct of folding the tree.

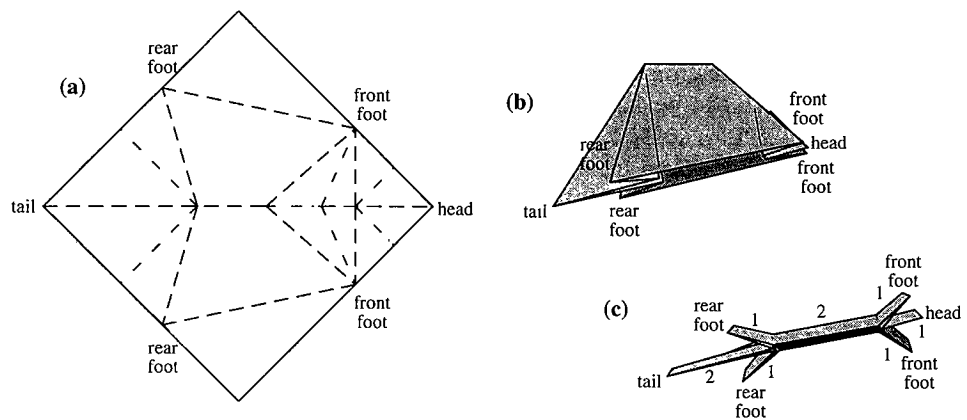


Figure 15: (a) Crease pattern for the lizard base. (b) The lizard base. (c) By repeatedly sinking the lizard base, you can even make a close approximation of the tree.

In fact, just as was the case with the circle method, it can be shown that any pattern of nodes you construct according to the rules of the tree method is foldable into the stick figure you started from. What we have here is a rudimentary algorithm for designing a large class of origami models. Any subject that can be approximated by a tree diagram — a stick figure — can be designed by (1) identifying the nodes, paths, and their lengths on the tree; (2) Laying out the nodes on a square such that the distances between any pair of nodes is larger than the corresponding path on the tree; (3) Folding the resulting pattern of nodes into a base. I don't wish to gloss over the great difficulty in step 3, of course. This algorithm gives a skeletal crease pattern, not a folding sequence. That you still have to find yourself. However, the search is always easier if you know for certain that the answer exists.

3 SYMMETRY AND THE TREE METHOD

If getting the right number and length of flaps were all that was needed to make a successful origami model, then the tree method algorithm as described above would be sufficient to do all origami design. However, nothing is perfect, and there are some limitations to the tree method – some fixable, some not. Although the pattern of nodes you get from the tree method is guaranteed to be foldable into a base with the right number and size of flaps, there is no guarantee that the folding method is (a) symmetric, (b) elegant, or (c) obvious. In fact, in many cases, the tree method, applied blindly, leads to wild, asymmetric patterns that don't give very nice-looking models.

For example, if we take a simple 5-node tree corresponding to a 4-legged creature, the most efficient skeleton puts the four nodes at the four corners of the square. This crease pattern can be folded into a base in several ways, but one of the most elegant is the traditional Bird Base as illustrated in Figure 16. The Bird Base is nicely symmetric, both from front-to-back and from side-to-side, and thus it can be used to fold animals that are bilaterally symmetric – animals whose left and right halves are mirror images of one another.

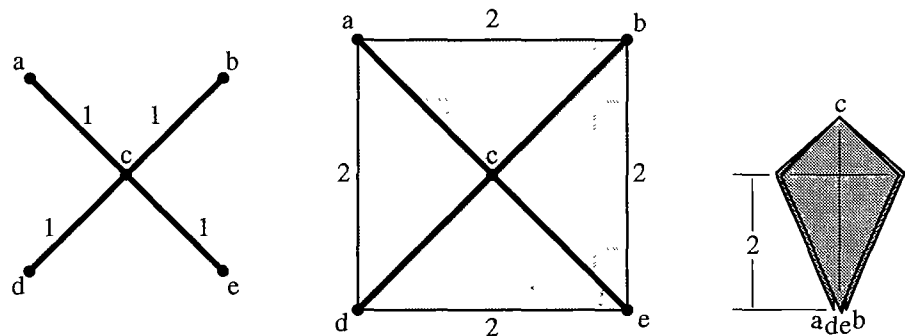


Figure 16: Tree, node pattern, and base for a base with four equal-length points emanating from a common point.

Now, suppose we want the same four flaps in our subject, but we want them separated by a segment (a body?) one unit long, as illustrated in Figure 17. If you place the nodes on a square and start figuring out path lengths, you are likely to find the pattern shown in Figure 17 (or one very similar) as an efficient placement of nodes.

The pattern in Figure 17 has a scale [scale=(1 unit of tree)/(side of square)] of $1/3=0.333$. However, this isn't the *most* efficient node pattern possible. The most efficient distribution of nodes is shown in Figure 18; it has a scale of $(\sqrt{14}-2)/5=0.348$, about 5% larger. This means that the base you fold from

Figure 18 would be 5% larger than the base folded from Figure 17 for the same size square, and thus each flap would contain proportionately less paper.

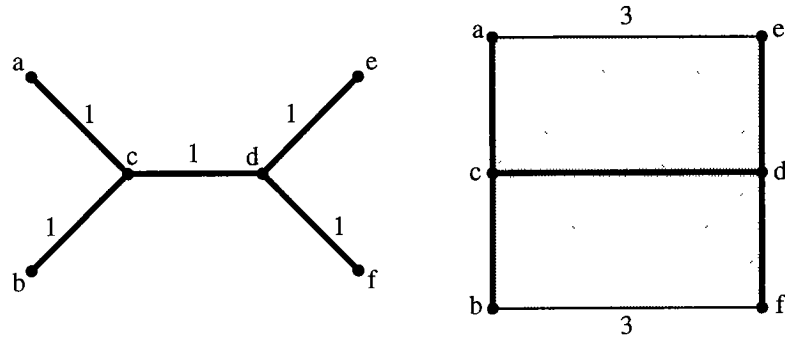


Figure 17: Tree and node pattern for a base with two pairs of equal-length points separated by a one-unit segment.

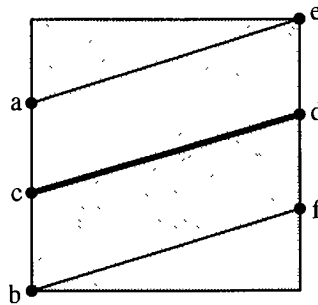


Figure 18: The most efficient node pattern for the tree of Figure 17.

Efficiency is not the same as elegance, however. Although Figure 18 is more efficient, Figure 17 makes a better origami animal (that is, if there is an animal out there with four appendages but no head or tail). Most animals have bilateral symmetry, but the crease pattern in Figure 18 does not. Figure 18 would be very difficult to fold into a symmetric-appearing base. Now, for most origami designers, starting from a bilaterally-symmetric crease pattern to fold a bilaterally-symmetric base would come instinctively. But if we are constructing an algorithm for design, then we must build symmetry into the algorithm explicitly. Although the most efficient crease pattern for the lizard came out symmetric, we will not always be so lucky. I have found that the node patterns for more complicated subjects – spiders, sea urchins, mating mosquitoes – *usually* are asymmetric and inelegant. Forcing symmetry of the crease pattern is an absolute requirement for an elegant design.

Of course, there are exceptions to every rule. Sometimes you are actually better off using an asymmetric base, either because your subject is inherently asymmetric, such as a fiddler crab, or because the asymmetry can be disguised, resulting in a

larger overall base. The exception probes the rule, but by definition is less common; so we need to enforce symmetry in most cases.

The symmetry shortfall is remedied by recognizing two special types of nodes – those that come in mirror-image pairs (like arms, legs, and feet) and those that lie along a symmetry plane (like the head, shoulders, hips, and tail). We will require that for bilaterally symmetric subjects, the nodes corresponding to mirror pairs of appendages must lie symmetrically about a symmetry line of the square. So, for the lizard, we would require that the node corresponding to the left rear foot be the mirror image of the node corresponding to the right rear foot and the node corresponding to the left front foot would be the mirror image of the node corresponding to the right front foot.

There are also nodes that correspond to body parts that lie along a line of symmetry of the subject, and we will simply require that these nodes lie on top of a line of symmetry of the square.

Note that a square has two different types of lines of mirror symmetry – one from corner to corner, and one from edge to edge. For the same tree structure and assignment of symmetry relationships, the two different choices of mirror symmetry will give different node patterns and consequently, different ways of folding the same subject. Figure 15 shows the crease pattern form a lizard when we use a diagonal line of symmetry; Figure 19 shows the crease pattern for a similar lizard, but based on a side-to-side line of symmetry. You might find it interesting to try to work out a folding sequence to transform the crease pattern in Figure 19 into a base.

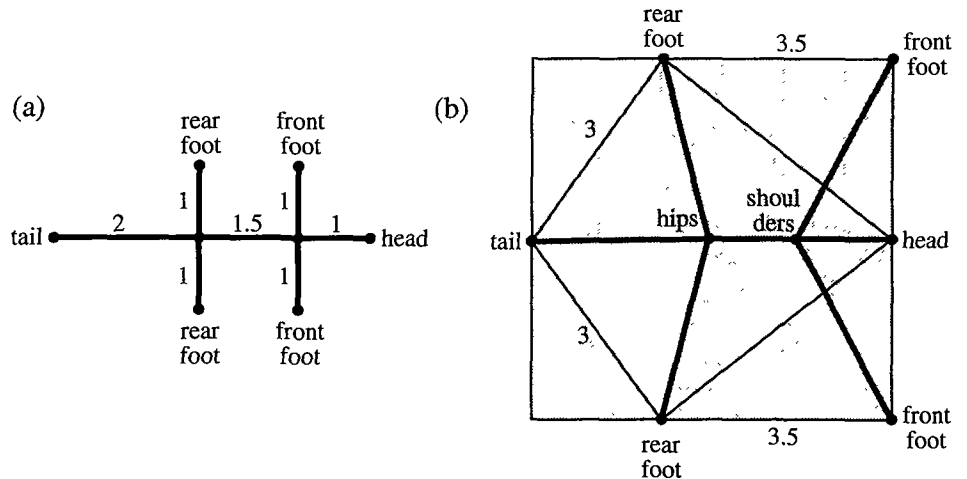


Figure 19: Tree and node pattern for the lizard with rectangular symmetry.

Two features of this crease pattern are worthy of note: one is that two of the corners go unused, which just goes to show you that even simple figure don't always use the corners. The other feature is that in Figure 19, the path from the front foot to the head is no longer a critical path as it was in Figure 15, which illustrates that as you change the orientation and/or lengths of paths, you need to carefully keep track of which paths are critical paths and which are not, because they can change.

I mentioned that the crease pattern you derive from the tree method is guaranteed to be foldable into a base with the same number and length of flaps as the original stick figure tree. However, this property is only true for infinitely thin (zero-width) flaps made with infinitely many creases. When you cut down on the number of creases to some finite number – and the jump from infinity down to six or eight is considerable – and let the width of the flaps become nonzero, you'll find that some flaps turn out shorter than they were in the tree. That is, some of the paper that might have gone into 'length' ends up going into 'width' instead. This situation happens whenever two flaps lie side-by-side, rather than one atop the other.

For example, if we were making a base for a four-legged animal with two legs on the left and two legs on the right and no body in between (like Figure 17, it's a *gedanken* animal), we would start with the tree and crease pattern shown in Figure 16, which would lead to a base very like the Bird Base, in which the four flaps come from the four corners of the square. According to our tree, each flap would then be as long as half the side of the original square, and indeed, each of the four flaps of a Bird Base is half as long as the side of the square from which it is folded.

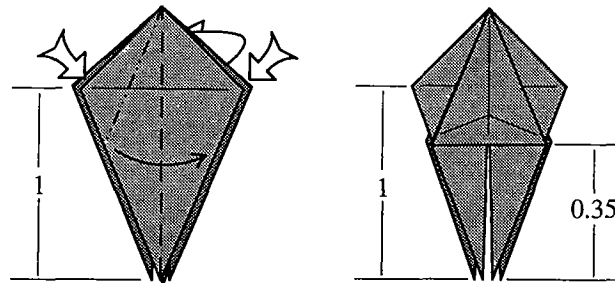


Figure 20: Making two symmetric pairs of flaps from a Bird Base.

Now, if we want our legs in two side-by-side pairs, the Bird Base doesn't quite work; it has two flaps side-by-side and the other two one atop the other. We can transform the four flaps into two pairs of mirror-symmetric flaps by spread-sinking two corners of the Bird Base, as shown in Figure 20. However, when we do this, we get an extra folded edge running across the two flaps that shortens the gap between them. Because the gap has been shortened, the 'free length' of the flaps has been reduced. Instead of being 0.5 times the side of a square, the flaps turn out to be only 0.354 of the side – a loss of about 30% of their length.

We can lengthen the gap and thus lengthen the flaps with a little more folding. If I put twice as many pleats running to the tips of the flaps, using the folding sequence shown in Figure 21, I can increase their length to 0.42, at the expense of doubling the number of layers, halving the width of the flaps, and adding a whole lot more folding. You can recover more of the gap by using more and more petal folds and narrower and narrower flaps, but only in the limit of infinitely many pleats (and an infinite number of layers) do you recover the full length of the flaps.

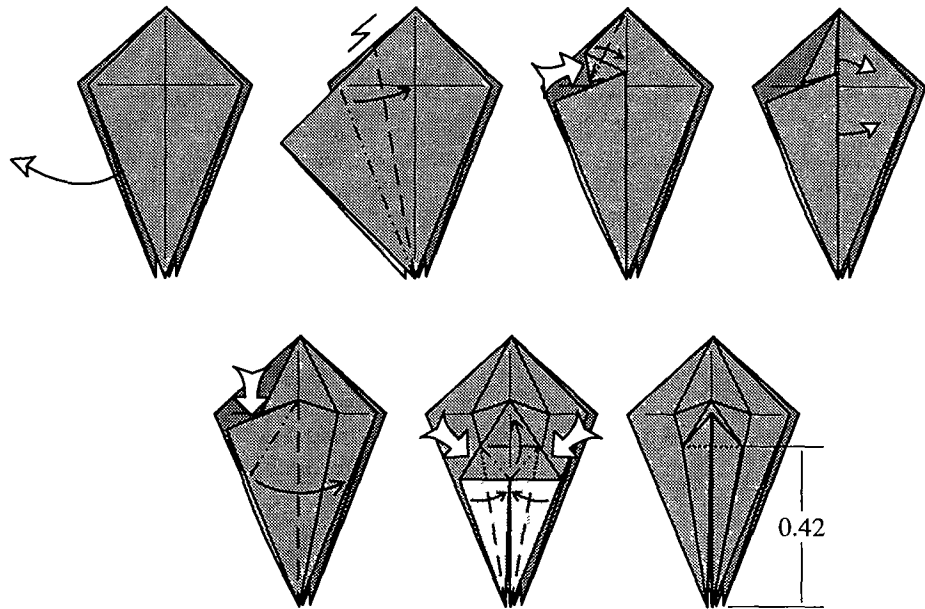


Figure 21: Making the gap deeper, at the expense of narrowing the flaps and introducing a lot more creases and layers.

This situation will always occur when you want two flaps to lie side-by-side, as opposed to one atop the other (note that the two flaps in each pair on one side of the Bird Base are separated from each another by the proper amount). The way to avoid having to do all of this back-and forth folding is for any pair of flaps that will wind up side-by-side, you add a little extra paper between them. In other words, you increase the minimum length associated with the path between that pair of flaps. The amount you must add depends on how many back-and-forth pleats you are willing to tolerate; typically, extending the path by about 40% will do the trick for a single back-and-forth pleat.

But, you're not ready to place paths yet. After you have placed all the nodes and lengthened some paths, you may find that paths that correspond to important creases are oriented at angles close to natural symmetry lines — i.e., at multiples of 30° or 22.5° — but aren't precisely at those symmetry lines. For esthetic reasons —

to avoid long, skinny crimps — it may be desirable to require those paths to lie exactly along the natural symmetry lines, even at the expense of a small decrease in efficiency. Thus, some of the nodes will have to be constrained to lie at certain angles with respect to one another.

Also for reasons of esthetics, you might want to insure that a particular node lands in a particular place on the square. For example, the most efficient crease pattern might put a particular node in the interior of the paper, whereas you might want the node to land on an edge or corner so that it has fewer layers.

4 A NUMERICAL IMPLEMENTATION OF THE TREE ALGORITHM

By the time you have drawn and labeled the tree, enumerated all of the paths and calculated their lengths, and noted which nodes need to lie symmetrically, and extended some of the paths, and forced some paths to run at certain angles and forced some nodes to lie on the edges of the square and so forth and so on, things can get pretty complicated. The biggest difficulty with applying the tree method of design is keeping track of all of the different paths and the constraints associated with each path. If the tree has N nodes, there are $N(N-1)/2$ different paths between pairs of nodes, each of which has a specific minimum length. The lizard, which has 8 nodes, has 28 paths! However, you really only need to keep track of the *critical* paths — the paths that are at their minimum length — and there are only six of those in the lizard.

The problem with complicated models is that as you try out different arrangements of the nodes, some paths stop being critical paths and other paths become critical paths. In addition, you must also keep track of which nodes lie symmetrically with respect to one another, and which paths you have extended the length of. This can get very difficult for a person to keep track of, and it is very easy to work out a design, only to discover that you overlooked a critical path somewhere in the criss-crossing network of paths that doesn't meet its minimum length requirement and throws your entire design off.

Ah, but what is difficult for a person to keep track of is a snap for a computer. The tree method of origami design that I have described above can be mathematically formalized and implemented as a computer algorithm. The algorithm is simple:

You do this to define the shape:

(1) Define a set of N nodes and the $N-1$ branches that connect them. Define the lengths of all of the branches. Give each node a set of coordinates on a square. (This is the setup input to the computer program.)

(2) Define a quantity called the 'scale', which is the length of a 1-unit branch on the square. Assign each node a pair of coordinates on the square. (This is an initial guess at the tree structure, which can be arbitrarily chosen.)

You do this to find the crease pattern:

(2) Construct all of the $N(N-1)/2$ possible paths between pairs of nodes and their lengths.

(3) Maximize the scale subject to the following constraints:

(a) The length of each path on the square must be greater than or equal to its length measured along the tree.

(b) The coordinates of all of the nodes must lie within the square.

(c) Nodes that lie on a symmetry line of the tree must lie on a symmetry line of the square.

(d) Nodes that are mirror-image pairs with each other on the tree must be mirror images of each other with respect to the symmetry line of the square.

(e) Paths close to natural symmetry lines of the square must lie exactly at the angles of those symmetry lines.

To computerize this algorithm, the verbal description of the algorithm needs to be converted to mathematical equations. I define x_i and y_i as the Cartesian coordinates of the i th node, define l_{ij} as the length (in units) of the path between nodes i and j as measured along the tree, define m as my scale factor, and define the coordinates of my square as lying between -1 and $+1$ on each axis. The problem becomes: Maximize m over the variables x_i and y_i subject to the constraints:

$$(a) \quad (x_i - x_j)^2 + (y_i - y_j)^2 \geq m^2 l_{ij}^2 \text{ for all } i, j$$

$$(b) \quad x_i \leq 1, x_i \geq -1, y_i \leq 1, y_i \geq -1 \text{ for all } i$$

$$(c) \quad y_i \cos \alpha - x_i \sin \alpha = 0, \text{ for each node } i \text{ that lies on a symmetry line at angle } \alpha \text{ with respect to the } x \text{ axis.}$$

$$(d) \quad (x_i - x_j) \cos \alpha - (y_i - y_j) \sin \alpha = 0, (x_i + x_j) \sin \alpha + (y_i + y_j) \cos \alpha = 0 \text{ for two nodes } i \text{ and } j \text{ that are symmetric about a line at angle } \alpha \text{ with respect to the } x \text{ axis.}$$

$$(e) \quad (y_i - y_j) \cos \alpha - (x_j - x_i) \sin \alpha = 0 \text{ for a path between two nodes } i \text{ and } j \text{ that lies at angle } \alpha \text{ with respect to the } x \text{ axis.}$$

This is a problem of constrained optimization. In fact, it would be a fairly simple problem in linear programming, if it weren't for the presence of the quadratic terms in letter (a) above. There is no algebraic solution to this system of equations, but the numerical techniques for the solution of such systems have been known for years. In the early 1970s, several workers contributed their efforts and subsequently their names to the development of a powerful technique for solving general nonlinear constrained optimizations, known as the Powell-Hestenes-Rockafeller Augmented Lagrangian Multiplier algorithm (Rockafeller, 1973).

Several months ago, I wrote a computer code that takes a description of a tree, constructs the equations in (a)–(e) above, solves for a local optimum, and displays the resulting skeletal crease pattern. The program is called TreeMaker, and it performs quite well. I've now used TreeMaker to design a number of non-trivial models, as well as to work out the examples presented in this series. Now admittedly, the lizard that I used throughout this series was a pretty simple model, and using a computer program to design it might seem like overkill. While simple models can be worked out entirely by hand, I have found that for complex subjects, the computer can find and evaluate efficient crease patterns much faster than I can by hand., including crease patterns that are not obvious to the unaided eye. In the next section, I'll illustrate the design process for a challenging model – an insect with legs, jaws, and antennae – and end with the final folding sequence for the design.

5 A CASE STUDY IN ORIGAMI DESIGN

I've always had a fascination for insects as origami subjects. For many years, most insects were considered too difficult to be realized in any but the most stylized form (or were realized with cuts and/or multiple sheets of paper). However, with the revolutions in modern origami design, insects have become commonplace. In origami, a six-legged, two-winged insect is no longer a challenge; it must have jaws, antenna, multiple color-changed wings, or something else to make it rise above the merely ordinary. Insects, with their skinny, jointed, multi-pointed bodies, are perfect candidates for the tree method – and thus, for computerized design.

The insect I chose to make for this article was a stag beetle, a beetle with six legs, jaws, and antennae, for a total of ten flaps. I decided to make the jaws and antennae somewhat shorter than the legs and to have them emanate from the head. I put the legs emanating from the thorax along with the abdomen, and make the abdomen somewhat longer than the legs, to give myself some extra paper to make a round body and/or pleated wings. This choice of subject also gives me an opportunity to compare man and machine; I already invented a stag beetle several years ago (it is included in my upcoming book, *Origami Insects* (Lang, 1994), and is shown in Figure 22. It will be interesting to see if TreeMaker can come up with a better design.

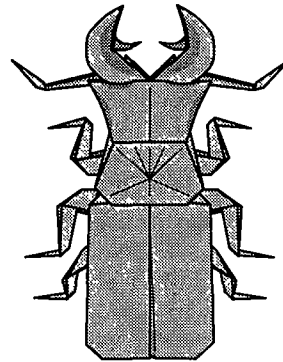


Figure 22: An origami stag beetle.

The first step in a design is to take the subject we are working from and convert it into a stick figure – the tree. Figure 23 shows a drawing of a real stag beetle and the tree that represents it. I have some choice in how I construct the tree. I've already specified that I want jaws and antennae, and of course I need six legs coming from the thorax. If I were being really daring, I could put separate wings on as well, but since beetles almost always keep their wings and elytra (forewings) folded over their body, we'll allocate a single flap to represent the wings and abdomen, and will plan on trying to suggest both features with pleats, crimps, and other detail folds.

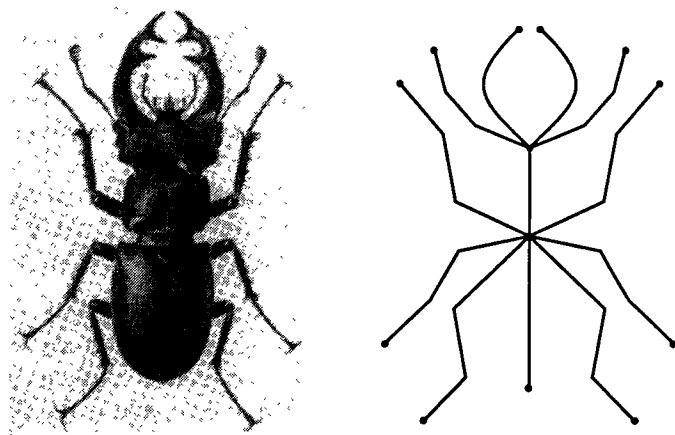


Figure 23: A real stag beetle and its equivalent stick figure, or 'tree'.

Now we need to assign lengths to the branches of the tree. On the real beetle, although each member of a pair of appendages are the same length, each pair – jaws, antennae, forelegs, midlegs, hind legs – is a slightly different length. When we

make an origami base, however, if all of the flaps are different lengths, then they'll all be slightly different widths where they join the body and each other. These differences translate into lots of little crimps (if you're careful) or little bodes (if you're not) to make the model lie flat; there will also undoubtedly be lots of misaligned edges. A base with lots of crimps, bodes, and misaligned edges looks messy, and a messy base makes a messy subject and a messy folding sequence. To keep the base and the folding sequence neat, it helps to make edges line up with each other whenever possible. Thus, we will stipulate that flaps that are *approximately* the same size on the subject should be *exactly* the same length on the base. For our stag beetle, we'll make the six legs all the same length and make the jaws and antenna the same length as each other but a little shorter than the legs.

We still need to quantify the lengths of the appendages. If we are trying to make an exact copy of the beetle in the photo, we could measure each leg, jaw, and antenna in the photo and assign that length to the corresponding branch of the tree, but that is a rather brute-force approach. Besides, it overlooks something: it is not enough just to find *some* folding sequence to make the base; we want to find an *elegant* folding sequence. 'Elegance' is a hard-to-define concept; it's easier to describe than to define. An elegant folding sequence is one in which all the folds flow naturally from one step to the next and the edges and creases line up with one another. An elegant model is one in which the lines are simple and clean. Elegant folding sequences arise when the creases and the base arise from natural symmetries of the paper. While TreeMaker will find the most efficient crease pattern for any given tree, it is up to the designer to pick the tree that best exploits the natural symmetries of the paper.

And one of the ways in which natural symmetries appear is in certain combinations of distances and lengths. Most origami – and in my opinion, the most elegant folding sequences – arise from exploitation of the symmetries associated with an angle of 22.5° , which is $1/16$ of a unit circle. This angle (and multiples thereof) show up repeatedly in elegant origami bases, as illustrated in Figure 24 in the four 'classic' bases (the Kite, Fish, Bird, and Frog Bases).

Along with this ubiquitous angle, there are ubiquitous distances, which are various algebraic combinations of integers and the number $\sqrt{2}$. If you unfold a model based on the symmetries of 22.5° and calculate the lengths of the major creases, you'll find that most of the distances work out to simple combinations of 1, 2, and $\sqrt{2}$, as shown in Figure 25.

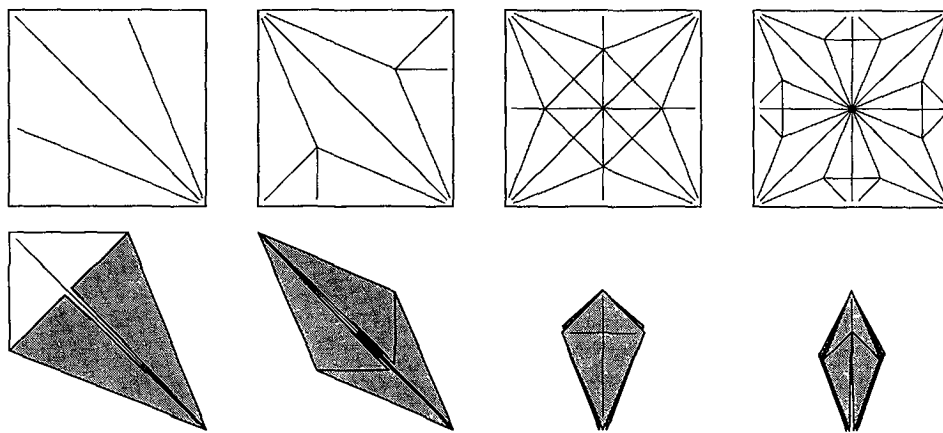


Figure 24: The four classic bases — Kite, Fish, Bird, and Frog base. The angle between any two creases is a multiple of 22.5° .

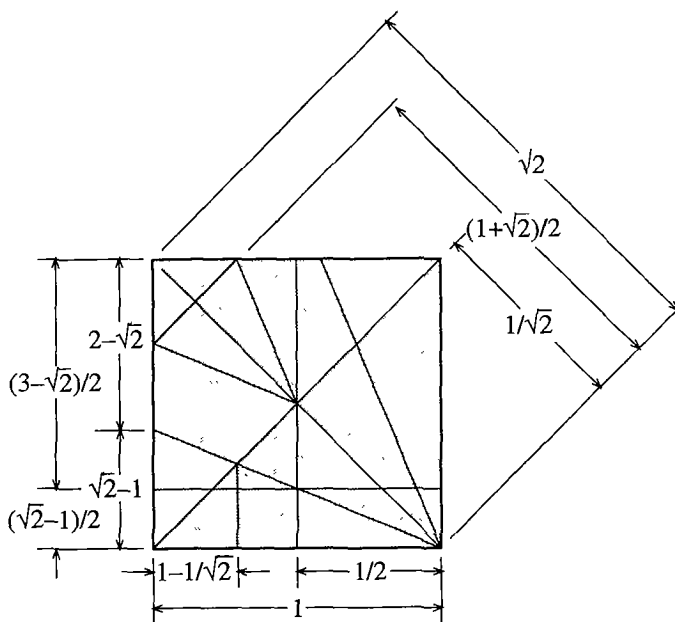


Figure 25: Typical distances to be found in crease patterns based on the 22.5° symmetry.

Since in symmetric, elegant crease patterns the same distances show up over and over, if we want our crease pattern to be symmetric and elegant, it makes sense to

use the distances that we know will already be present. Conversely, if we use strange values for distances, values that don't correspond to 22.5° symmetries, it's going to be really hard to find a folding method that results in an elegant base. We are more likely to find a symmetric base if we express all the dimensions in terms of 'symmetric' distances.

This is really not so hard to do, because there are a lot of symmetric distances to choose from. It turns out that for any ideal distance, there is an symmetric distance with nearly the same value. For example, Figure 26 displays the algebraic form of all the distances shown in Figure 25, along with their decimal value and for comparison, a line whose length is proportional to that value. No matter what length a given flap or appendage is, you can find a symmetric distance that differs from the target by only a few percent. Figure 26 is an incomplete list of symmetric distances associated with the 22.5° symmetries – there is a whole other family of distances associated with multiples of 15° – but you can extend this list to larger and smaller values simply by multiplying or dividing by factors of 2. Any numerical value you choose lies fairly close to one of these significant lengths.











Algebraic	Decimal	Linear
$\sqrt{2}$	1.414	
$(1+\sqrt{2})/2$	1.207	
1	1.000	
$(3-\sqrt{2})/2$	0.793	
$1/\sqrt{2}$	0.707	
$2-\sqrt{2}$	0.586	
1/2	0.500	
$\sqrt{2}-1$	0.414	
$1-1/\sqrt{2}$	0.293	
$(\sqrt{2}-1)/2$	0.207	

Figure 26: Distances found in the eightfold (22.5°) symmetry. The first column is the algebraic formula for a length; the second is its decimal value; and the third shows a line segment proportional to the length.

So for our origami stag beetle, we will choose flap lengths that correspond to interesting lengths. We will define our legs to be 1 unit long each; the jaws and antenna are slightly smaller, so we'll choose a length of $1/\sqrt{2}=0.707$ for those. The abdomen should be somewhat longer than the legs, to give us some extra paper for crimps and pleats to form the wings and/or body segments; we'll make it $\sqrt{2}=1.414$ units long. Finally, we need a short segment between the thorax and the head, so that the jaws and antennae don't come from the same part of the model as the six legs (otherwise, we could just use the circle method). This short segment will be

$\sqrt{2}-1=0.414$ units long. We now have completely defined the target tree for our design efforts.

Now that we have defined our tree, we could start enumerating paths and calculating their lengths, but since our tree has 13 nodes, there are 78 possible paths between nodes that must be considered, which is a lot to try to keep track of. So I will turn now to describe the use of my TreeMaker computer program. TreeMaker has a point-and-click user interface that allows you to enter a tree structure by clicking on a square to create nodes and branches, clicking and dragging to shift them around, and double-clicking on nodes and branches to change their properties. The tree as I entered it is shown in Figure 27. TreeMaker allows you to name nodes with the part of the subject they correspond to and displays the lengths of branches, so I've shown those numbers and names for clarity.

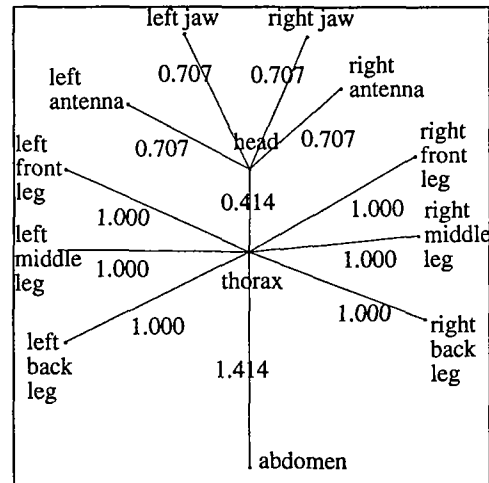


Figure 27: The tree for a stag beetle, superimposed over a square. The numbers are the desired lengths of the branches; the names identify which part of the subject corresponds to each node.

To give you an idea of the complexity of this model that has 13 nodes and 78 paths, I've shown all 78 paths in Figure 28, each represented by a straight line. Every line corresponds to a constraint on the distance between two nodes. Now, if you or I were working this design out by hand, we would figure out which paths are the important ones and which could be ignored, but the computer is not so clever, and needs to consider them all.

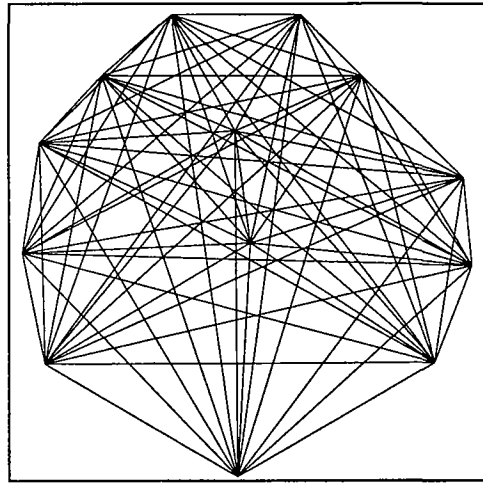


Figure 28: Every possible path between two nodes is represented by a line. There are 78 total paths between the 13 nodes.

Once we have set up nodes and branches, the computer calculates all possible paths and the minimum length of each path. To find a possible arrangement of nodes, the computer needs to optimize (find the maximum value of) the scale of the tree while insuring that all paths meet their minimum distance constraints. At this point, we could just tell the computer to go through its optimization process. TreeMaker arrives at the following arrangement of nodes:

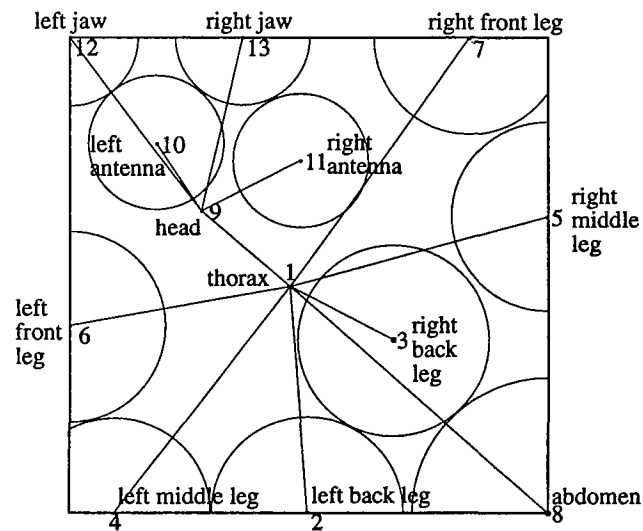


Figure 29: A node pattern that satisfies all of the path constraints for the tree. The circles identify the terminal nodes of the tree. The scale is 0.2012.

This first go-around was certainly less than successful. One back leg is a middle flap; the other is an edge flap. One jaw is a corner flap; the other is an edge flap. The head is twisted way around to the right. Folding this crease pattern into a base would be a nightmare. About the only positive thing I can say is that the scale – the length of one unit compared to the size of the square – is a very respectable 0.2012, meaning that the length of a leg is about 1/5 of the length of the side of the square. Nevertheless, there is no symmetry, no rhyme or reason, and most importantly, no easy or obvious way to fold this crease pattern into a base! So the first foray into computerized design is without doubt, a failure.

Well, maybe not quite a complete failure. We haven't yet imposed any symmetry requirements, and lack of symmetry is the main problem with the node pattern in Figure 29. We need the legs, jaws and antennae to be mirror images of each other with respect to a symmetry line of the square – and they aren't in Figure 29. We also need the head, thorax, and abdomen to lie directly on top of the symmetry line, since they lie on the symmetry line of the subject. Since the shape in Figure 28 is oriented roughly along the diagonal, let's choose the diagonal of the square to be our line of symmetry. TreeMaker allows you to select either a diagonal or a rectangular line of mirror symmetry for the square, to constrain individual nodes to lie on the symmetry line and to force pairs of nodes to be mirror images of each other about the symmetry line. So, when we establish these symmetry requirements and restart the optimization, we arrive at the node pattern shown in Figure 30. This pattern has a scale of 0.1853. The scale has decreased by about 8%, which means that the base folded from this crease pattern will be about 8% smaller than the base folded from the crease pattern in Figure 28. But the pattern in Figure 30 is more likely to be foldable into a *symmetric* base and thus a symmetric model, and so the small decrease in scale is an acceptable price to pay.

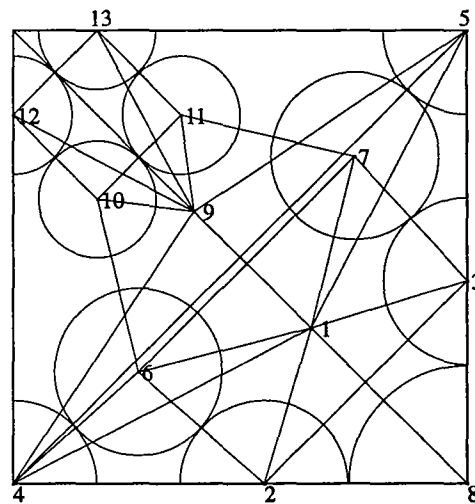


Figure 30: Optimized node pattern for the stag beetle with mirror symmetry invoked. The scale is 0.1853.

There is one other chore before us; if we wish our finished base to be flattened from top to bottom (as is the beetle in Figure 22), then the legs come in side-by-side pairs. To avoid having to make infinitely many pleats between them, we need to allocate some extra paper by extending the paths between mirror pairs of legs and jaws. To keep the back-and-forth folding to a minimum, I'd like to use a single back-and-forth pleat between the legs, such as is accomplished in the Frog Base by petal-folding an edge. For this petal fold, the distance between the two flaps must be $\sqrt{2}$ times as long as the minimum path, so between side-by-side pairs of legs, I'll need to extend each path by a factor $\sqrt{2}$. Just as you could edit individual nodes, TreeMaker lets you edit individual paths and to increase the minimum separation between node pairs. We do this for each pair of mirror flaps, increasing the separation between the jaws from 1.414 to 2.000, between the antennae from 1.414 to 2.000, and between pairs of legs from 2.000 to 2.828. After bumping up these paths, I re-optimize. The scale has decreased from 0.1853 to 0.1775, which is still not a large decrease, but the resulting pattern, shown in Figure 31 is *now* closer to something that can conceivably be folded into an elegant base.

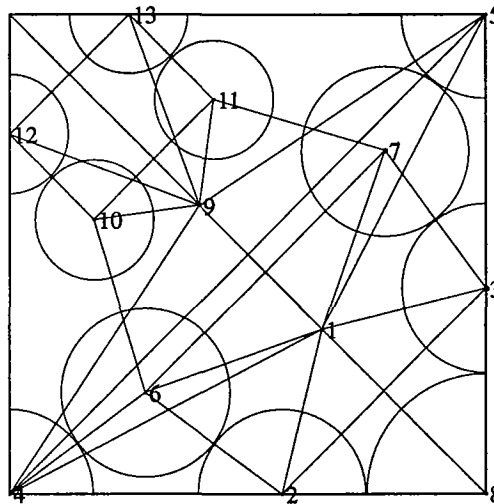


Figure 31: Optimized node pattern for the stage beetle with mirror symmetry and path extensions between side-by-side flaps. The scale is 0.1775.

But wait – there’s more. The crease pattern shown in Figure 31 can *in principle* be folded into a beetle, but I doubt that the folding sequence is very clean or elegant because the major crease lines are running at irregular angles. Not only do we want our distances to be symmetric; we want the angles of major creases also to lie at multiples of 22.5° whenever possible. Thus, we need to set some more constraints that force paths to lie at multiples of 22.5° . We can’t force all the paths to the same angles, though; just the ones that correspond to major creases of the base.

Forcing paths to lie at particular angles is another capability of TreeMaker. For any given path, there are 8 possible multiples of 22.5° to set the angle to, but you might guess that you'll perturb the crease pattern the least by looking for paths between adjacent nodes that are nearly at multiples of 22.5° and setting them to the nearest multiple. If I do this for the paths marked in Figure 31 and again re-optimize, I find the following crease pattern in Figure 32, which is more nicely symmetric, and which has a scale of 0.1726.

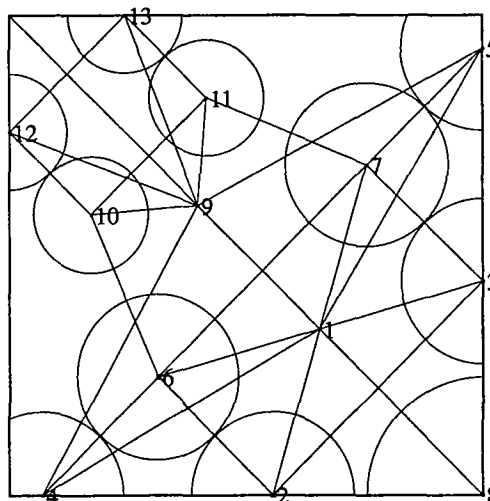


Figure 32: Optimized node pattern for the stag beetle with mirror symmetry, path extensions, and angular constraints. The scale is 0.1726.

It's interesting; each time we add some more constraints, the scale, and thus the size of the model, gets reduced. That is, we are trading off folding *efficiency* for folding *elegance*. This is an esthetic judgment, and requires the active participation of the designer. Computer programs like TreeMaker can simplify aspects of origami design, but they are no substitute for good design sense.

In fact, it might even be considered a bit of a stretch to say that TreeMaker is 'designing' the base. It isn't so much creating a useful arrangement of nodes as it is eliminating the enormous number of useless ones. For 13 nodes, there are 27 degrees of freedom in the placement of the nodes on the square (two coordinates for each node plus the scale). A given configuration of nodes and scale can be described by a single point in a 27-dimensional space. The various constraints on path distances and angles can be written as equations that subdivide the 27-dimensional space into spheroidal regions of feasibility (for inequalities) or spheroidal surfaces (for equalities). The combination of all of the equations define an incredibly convoluted hyperdimensional blob of feasible space over which the scale – the merit function – varies in value, and all we are doing is looking for

those flaps with the maximum value of the scale. The convolution of the surface implies that there are numerous local minima on the surface and there is no guaranteed routine for finding a global minimum. Just because we found the best configuration of nodes from a given starting point, a different starting point might give us an even more efficient arrangement of nodes!

So, one new starting point we might try is to move the abdomen into the center of the paper, rather than using a corner for the abdomen. This would have the effect of moving the legs closer to the corner opposite the head, but exactly what else is not clear until we re-optimize TreeMaker. This starting point gives the crease pattern shown in Figure 33, which has a slightly larger scale of 0.1745, and would give a barely larger base than the pattern of Figure 22. Significantly, the crease pattern to transform Figure 33 into a base would in all likelihood be entirely different from the pattern for Figure 22! Thus, simply by starting from a different initial configuration, you can discover entirely new ways of folding existing subjects simply by moving the initial distribution of nodes around.

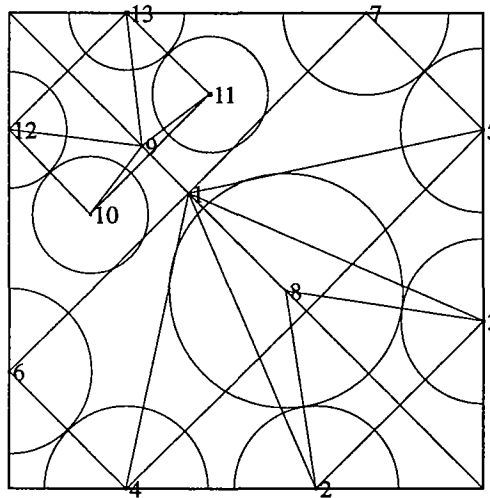


Figure 33: Node pattern with the abdomen in the center. The scale is 0.1745.

Figure 33 has one other nice feature compared to Figure 32; all of the legs come out as edge flaps in Figure 33, whereas two of the legs are middle flaps in Figure 32. For an insect, we want the legs to be as thin as possible, and the extra thickness in a middle flap might be something to be avoided. (Then again, it might not; I know a lot of good insect designs that use middle flaps for legs.) If we start with Figure 32, force all of the legs to lie on the boundary of the square and set angles that are near 45° to exactly 45° , we get the crease pattern shown in Figure 34, which has a scale of 0.1715, still smaller than any of the preceding patterns, but acceptable.

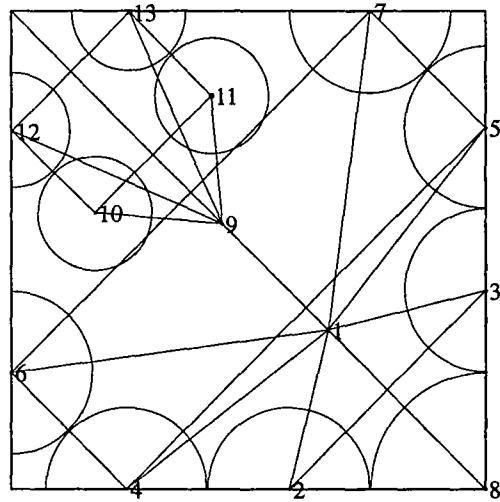


Figure 34: Node pattern for the stag beetle with all legs constrained to lie on the edge of the square. The scale is 0.1715.

We might also wish the jaws to be edge flaps and allow the legs to be middle flaps. This constraint, plus a few angular constraints gives the very symmetric crease pattern shown in Figure 35. Not only are all the major creases at 45° , but the center of the paper is an intersection of two important creases. The scale has decreased further, to 0.167, almost precisely $1/6$ of the side of the square.

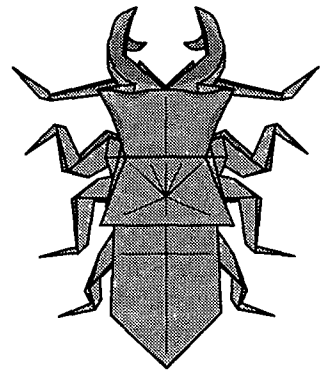
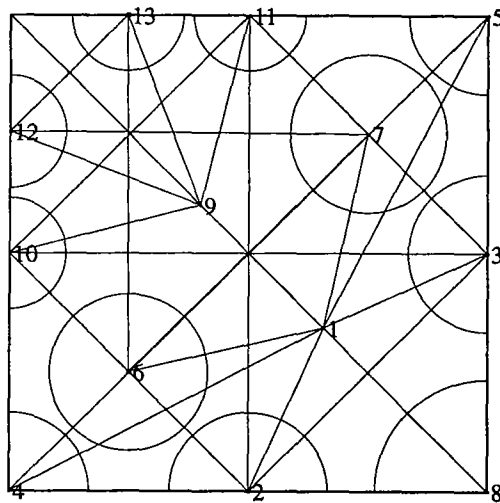


Figure 35: (Left) Node pattern for the stag beetle with all jaws constrained to lie on the edge and with the antenna and two legs on each side constrained to a 45° line. The scale is 0.1665. (Right) Resulting stag beetle.

In fact, to my taste, Figure 35 is too symmetric. For years, origami designs were made from bases such as the Bird and Frog base that are four-fold symmetric about the center of the square, which always resulted in a single thick flap somewhere in the model – a waste of paper, and a waste of symmetry. Only in rare circumstance does a four-fold symmetric base make a good match to a two-fold symmetric subject. The crease pattern shown in Figure 35 can be folded into the rather stubby stag beetle shown in Figure 35 – you might wish to try to work out a folding sequence – but the sequence I found is rather predictable, and is four-fold symmetric for much of the sequence.

Of course, we could try something completely different by trying the alternate line of symmetry – that is, orienting the model along an axis parallel to the side of the square, rather than along the diagonal. Setting this line as our symmetry line, optimizing, and forcing a few major crease lines to verticals results in the following intriguing crease pattern, which clearly has a 30° symmetry built in. It also has a scale of 0.1808, the largest of any of the crease patterns we have examined so far.

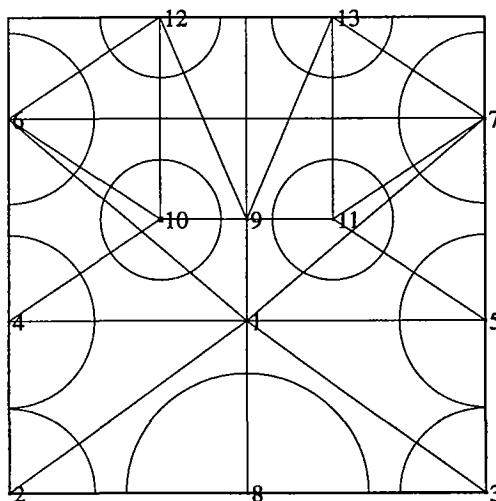


Figure 36: Node pattern for the stag beetle with rectangular symmetry and path angle constraints. The scale is 0.1808.

Now, *all* of these crease patterns can be folded into a stag beetle, and the most interesting folding sequence for each is going to be very different from the sequence for any of the other crease patterns. So right here we have the beginnings of five or six new models. Of course, the question of how you develop a folding sequence that takes a crease pattern to a base is no small matter, and is in fact worthy of an entire set of articles in its own right. Setting aside the issue of how you fold up a crease pattern into the base, though, you see the boundless possibilities that you can avail yourself up even from a single tree structure! I've developed stag

beetles from about half of the crease patterns I've shown here. I'll close this article with my favorite, which is derived from Figure 34, and is illustrated in Figure 37.

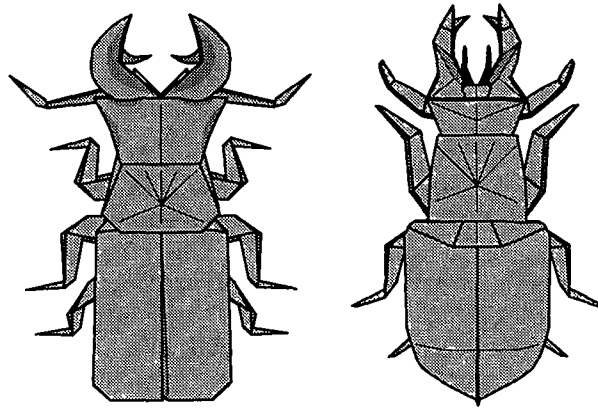


Figure 37: Stag beetles, according to (left) human, (right) computer.

You might wish to try to find your own folding sequence for one of these crease patterns. Better yet – change some dimensions of the tree and start over yourself! Give the beetle extra-long jaws this time, or get rid of the antennae, or add a pair of open wings. Whatever you do, you are bound to find new and undiscovered models lurking in the paper. Whether you use TreeMaker, write your own computer program, or work out your design by hand, mathematical origami design is a powerful technique that can lead you to new vistas of origami design.

REFERENCES:

- Engel, P. (1989) *Folding the Universe*, New York: Random House.
- Gardner, M. (1992) *Fractal Music, Hypercards, and More*, New York: Freeman.
- Kasahara, K. (1988) *Origami Omnibus*, New York: Japan Publications.
- Lang, R. J. 1994) *Origami Insects*, New York: Dover Publications, (to be published.)
- Montroll, J. (1985) *Animal Origami for the Enthusiast*, New York: Dover.
- Rockafeller, R. T. (1973) A dual approach to solving nonlinear programming problems by unconstrained optimization, *Mathematical Programming*, 5, 354-373.