0 Introduction

The topic of this book is the geometry of folding and unfolding, with a specific emphasis on algorithmic or computational aspects. We have partitioned the material into three parts, depending on what is being folded or unfolded: linkages (Part I, p. 7–164), paper (Part II, p. 165–296), and polyhedra (Part III, p. 297–441). Very crudely, one can view these parts as focusing on one-dimensional (1D) objects (linkages), 2D objects (paper), or 3D objects (polyhedra). The 1D–2D–3D view is neither strictly accurate nor strictly followed in the book, but it serves to place related material nearby.

One might classify according to the process. Folding starts with some unorganized generic state and ends with a more structured terminal "folded state." Unfolding is the reverse process, but the distinction is not always so clear. Certainly we unfold polyhedra and we fold paper to create origami, but often it is more useful to view both processes as instances of "reconfiguration" between two states.

Another possible classification concentrates on the problems rather than the objects or the processes. A rough distinction may be drawn between *design* problems—given a specific folded state, design a way to fold to that state, and *foldability* questions—can this type of object fold to some general class of folded states. Although this classification is often a Procrustean bed, we follow it below to preview specific problem instances, providing two back-to-back minitours through the book's 1D–2D–3D organization. We make no attempt here to give precise definitions or state all the results. Our goal is to select nuggets characteristic of the material to be presented later in detail. The first pass through (design problems) emphasizes geometry, the second pass (foldability questions) emphasizes computational complexity.

0.1 DESIGN PROBLEMS

0.1.1 I: Kempe Universality

A planar *linkage* is a collection of fixed-length, one-dimensional segments lying in a plane, joined at their endpoints to form a connected graph. The joints permit full 360° rotation, and the rigid segments are permitted to pass through one another freely. With one or more joints pinned to the plane, the motion of any particular free joint J is constrained by the structure of the linkage. A specific question here is this:

> Let S be an arbitrary algebraic curve in the plane. Is there some linkage so that the motion of some free joint J traces out precisely S?

Figure 0.1. There is a linkage that traces a thin version of this collection of curves.

The surprising answer is YES, even if the "curve" includes cusps and multiple pieces. Thus, there is linkage that "signs your name" (Figure 0.1).

The original idea for the construction of a linkage to trace a given curve is due to Kempe from 1876, but technical difficulties in the proof were not completely resolved until 2002. We discuss the history, sketch Kempe's beautiful but flawed proof, and the recent repairs in Section 3.2. An interesting open question here (Open Problem 3.2) is whether there is a linkage to follow any algebraic curve that does not self-intersect during the motion.

0.1.2 II: Origami Design

The epitome of a folding design question is provided by origami:

> Given a 3D shape (an origami final folded state), find a crease pattern and sequence of folds to create the origami (if possible) from a given square piece of paper.

Stated in this generality, this problem remains unsolved, which is one reason that origami is an art. However, in practice origami shapes are a subset of all possible 3D shapes. They are those shapes constructible in two steps: creating an origami *base*, and creasing and adjusting the remaining paper to achieve the desired design. An origami base can be considered a *metric tree*: a tree with lengths assigned to the edges. Creating the base is the hard part of origami design; the secondary adjusting steps are relatively easy (for origami masters).

In the past decade, Robert Lang has developed an algorithm to construct a crease pattern to achieve any given "uniaxial base," the most useful type of origami base. This represents a huge advance in origami design, which previously relied on just a handful of known origami bases. It has allowed Lang to design amazingly intricate origami, such as the mule deer shown in Figure 0.2.

We describe his algorithm, implemented in a program he calls TreeMaker, in Chapter 16. One issue remaining here is that although there is strong experimental evidence that the crease pattern output by TreeMaker leads to a non-self-intersecting folded state, this has yet to be formally proved.

0.1.3 III: Unfolding to Net

The oldest problem we discuss in this book goes back (in some sense) to Albrecht Dürer in the sixteenth century, who drew many convex polyhedra cut open along edges and unfolded flat to a single nonoverlapping piece, now called a *net*. See Figure 0.3 for an example from his *Painter's Manual* (1525).

It remains unresolved today (Open Problem 21.1) whether this is always possible:

> Can the surface of every convex polyhedron be cut along edges and unfolded to a net?

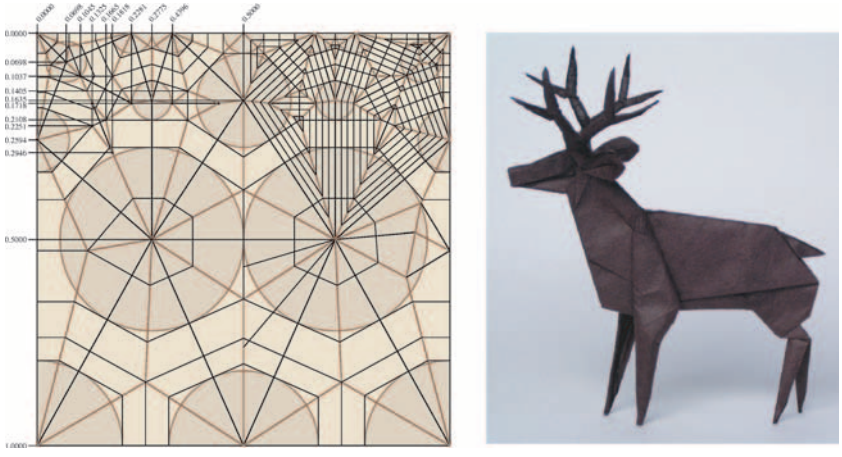

Figure 0.2. A "mule deer" folded by Robert Lang (opus 421): http://www.langorigami.com/art/gallery/gallery.php4?name=mule_deer The crease pattern was designed using TreeMaker.

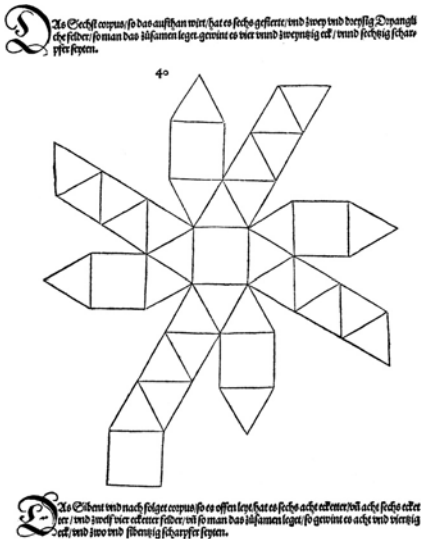

Figure 0.3. A net for the snub cube, drawn by Dürer.

Here the design target is not a specific shape, but rather any planar shape that avoids overlap. We discuss in Chapter 22 evidence for and against the conjecture that the answer to this question is YES, and spin off in a number of related directions. One such question is obtained by removing the *edge-unfolding* restriction: that the cuts must be along edges of the polyhedron. For unrestricted cuts, the answer to the posed question is known to be YES (Section 24.3).

0.2 FOLDABILITY QUESTIONS

We now make our second pass through the three parts of the book, this time concentrating on foldability.

0.2.1 I: Ruler Folding

A polygonal *chain* is a linkage whose graph is just a path. If one views it as a "carpenter's ruler," it is natural to seek to fold it up into as compact a package as possible. This is known as the "ruler folding problem":

> Given a polygonal chain with specific given (integer) lengths for its n links, and an integer L, can it be folded flat (each joint angle either 0 or 180°) so that its total length is $\leq L$?

Of course, sometimes the answer is YES and sometimes NO, depending on the link lengths and L. What is interesting here is the "computational complexity" of deciding which is the case. It was proved in 1985 that answering this question is difficult: NP-complete in the technical jargon. Effectively this means that for, say, $n = 100$, it is not feasible to decide. We present the (easy) proof in Section 2.2.2, one of several proofs throughout the book that establish similar NP-completeness results.

0.2.2 II: Map Folding

A *flat folding* of a piece of paper is a folding by creases into a multilayered but planar shape. The paper is permitted to touch but not penetrate itself. A fundamental question on flat folding is this:

> Given a (rectangular) piece of paper marked with creases, with each subsegment marked as either a mountain or valley crease, does it have a flat folded state?

It was proved in 1996 that answering this question is NP-hard, which means at least as intractable as an NP-complete problem. We present a (complex) proof in Section 13.2 for the easier case when the mountain—valley assignments are not given. When they are specified, the proof is even more difficult. An interesting variant remains open (Open Problem 14.1):

> Given a (rectangular) piece of paper marked by a regular square grid of unit-separated creases, with each subsegment marked as either a mountain or a valley crease, can it be folded into a single 1×1 square?

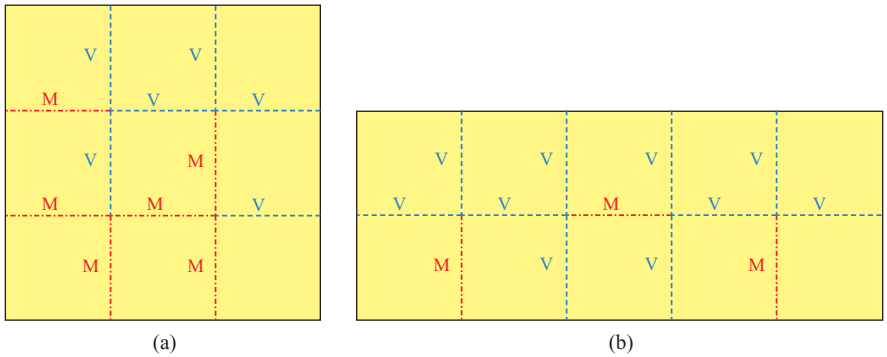

Figure 0.4. (a) A 3×3 map that can be folded (but not easily!) (cf. Figure 14.1); (b) a 2×5 map that cannot be folded (Justin 1994).

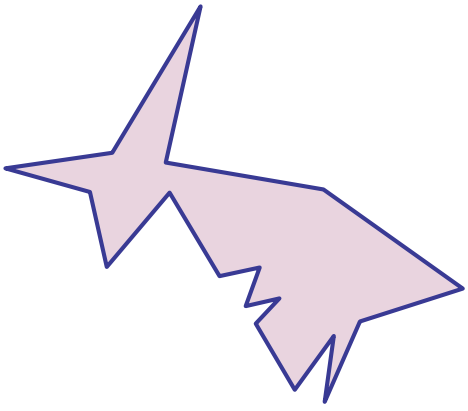

Figure 0.5. A polygon that folds to a convex polyhedron of 16 triangular faces.

See Figure 0.4 for examples. The added constraints on the creases in the map-folding problem may make this tractable even though the unconstrained problem is not.

0.2.3 III: Polygon Folding

The inverse of unfolding a convex polyhedron to a net is folding a polygon to a convex polyhedron:

> Given a polygon of n vertices, can it fold to some convex polyhedron?

Here we assume that the folded polygon covers the surface of the polyhedron precisely once: there are neither gaps in coverage nor double coverage. Nor are we insisting that the polygon be an edge unfolding of the polyhedron; rather, the cuts that produce it are arbitrary.

Figure 0.5 shows an example that can fold. Not all polygons can—there are "unfoldable polygons"—so the question makes sense. It was first settled for a special case, when the folding is restricted to glue whole edges of the polygon to one another. Then an $O(n^3)$-time algorithm is known for a polygon with n vertices. Without the edge-to-edge restriction, only an exponential-time algorithm is available. Both algorithms are discussed in Section 25.1. It remains open (Open Problem 25.3) whether the question can be decided in polynomial time.

These quick tours give some sense of the material, but there is no substitute for plunging into the details, which we now proceed to do.

PART I

Linkages

1 Problem Classification and Examples

Our focus in this first part is on one-dimensional (1D) linkages, and mostly on especially simple linkages we call "chains." Linkages are useful models for robot arms and for folding proteins; these and other applications will be detailed in Section 1.2. After defining linkages and setting some terminology, we quickly review the contents of this first part.

Linkage definitions. A *linkage* is a collection of fixed-length 1D segments joined at their endpoints to form a graph. A segment endpoint is also called a *vertex*. The segments are often called *links* or *bars*, and the shared endpoints are called *joints* or *vertices*.[1] The bars correspond to graph edges and the joints to graph nodes. Some joints may be *pinned* to be fixed to specific locations. Although telescoping links and sliding joints are of considerable interest in mechanics, we only explore fixed-length links and joints fixed at endpoints. (We'll use the term *mechanism* to loosely indicate any collection of rigid bodies connected by joints, hinges, sliders, etc.) An example of a linkage is shown in Figure 1.1.

Overview. After classifying problems in this chapter, we turn to presenting some of the basic upper and lower complexity bounds obtained in the past 20 years in Chapter 2. We then explore in Chapter 3 classical mechanisms, particularly the pursuit of straight-line linkage motion. In contrast to these linkages, whose whole purpose is motion, we next study in Chapter 4 when a linkage is *rigid*, that is, can move at all. Most of the remainder of Part I concentrates on chains, starting with reconfiguring chains under various constraints in Chapter 5. We then reach in Chapter 6 what has been the driving concern in the community for at least a decade: deciding under what condition a chain is *locked*. In the language of configuration spaces (to be introduced shortly), rigidity corresponds to an isolated point in configuration space and lockedness to a disconnected component in configuration space. This leads naturally to interlocked collections of chains in Chapter 7. We close with a study of fixed-angle chains in Chapter 8, which leads directly to protein folding in Chapter 9.

We now turn to classifying the problems pursued in later chapters.

[1] Sometimes it is convenient to place an endpoint joint of one link in the interior of another rigid link. This structure can always be simulated by links that only share endpoint joints by adding extra links to ensure rigidity, as we will show later (Figure 3.10). So we sometimes will use interior joints in figures, but in our analysis assume all joints are mutual endpoints.

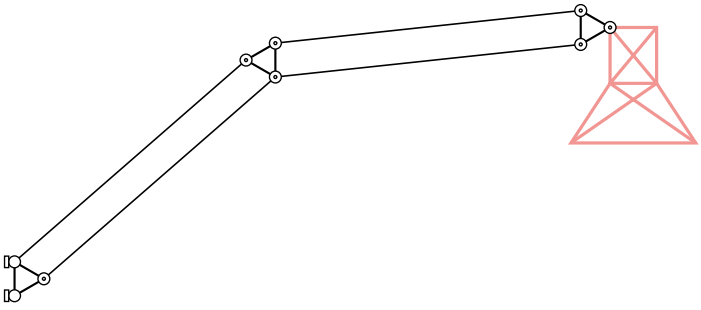

Figure 1.1. A linkage. The circles represent joints at which turning is possible; the two leftmost joints are pinned to the plane. The shaded "lamp" structure is rigid (because of the interior diagonals).

1.1 CLASSIFICATION

There are six features or "parameters" which together classify most of the material we discuss in Part I. These features fall into two groups: those classifying the linkages and those classifying the problems studied. We now quickly survey this classification, with many details deferred to later sections.

First, linkages can be distinguished according to their graph structure, the dimension, and intersection conditions. The graph structure may be a general graph (e.g., Figure 1.1), a tree, a single cycle, or a simple path. A linkage whose graph is a cycle is called a *polygon*; one whose graph is a path we will call a *polygonal chain* or just a *chain*. In the literature a chain is also called an *arc*, a *robot arm*, or an *arm*.

The second parameter is the dimension of the ambient space in which the linkage lives: two, three, four, or higher dimensions. We abbreviate these as 2D, 3D, 4D, and kD, and often use *planar* to mean 2D.

The third parameter classifies how the linkage may intersect itself or intersect obstacles in its environment. With no constraints and no obstacles, the linkage may freely pass through itself, with all joints permitting arbitrary rotation. For 2D chains, this model can be realized with the links at slightly different levels parallel to a plane, with joints realized as short pegs perpendicular to that plane. An intermediate situation of some interest is a linkage that can freely pass through itself, but which is forbidden to penetrate certain fixed obstacles in its environment. This is especially useful for modeling workspace restrictions for robot arms. Finally, much work has assumed an obstacle-free environment but insists that the linkage never self-intersect. A linkage which does not self-intersect is (confusingly) said to be *simple* in the literature, so often this restriction is phrased as demanding the *simplicity* of the linkage. For example, a polygon that does not self-intersect is a *simple polygon*. There is a somewhat subtle distinction between self-intersection and self-crossing, which we will revisit later (in Section 6.8.2).

The foregoing three parameters classify the linkages. We now turn to classifying the questions asked. There are again three primary features that distinguish the questions: the geometric issue, the answer desired, and the type of complexity bound sought.

The most specific geometric problem is "reconfiguration." A *configuration* of a linkage is a specification of the location of all the link endpoints (and therefore of the

link orientations and joint angles).[2] A configuration respects the lengths and non-self-intersection of the linkage (if so specified), but may penetrate obstacles. A configuration which avoids all obstacles is said to be *free*, and one that touches but does not penetrate obstacles is *semifree*.

For example, a configuration of an n-vertex polygon in 3D may be specified by $3n$ numbers: n triples of vertex coordinates. The *configuration space* is the space of all configurations of a linkage.[3] In the polygon example, this space is a subset of $3n$-dimensional space, $\mathbb{R}^{3n}$. The *reconfiguration* problem asks, given an initial configuration $\mathcal{A}$ and a final configuration $\mathcal{B}$, can the linkage be continuously reconfigured from $\mathcal{A}$ to $\mathcal{B}$, keeping all links rigid (their original length), staying within the ambient space (e.g., a 2D plane), without violating any imposed intersection conditions? (When a linkage satisfies all the conditions it is said to be in a *legal* configuration.) A slightly less specific problem is to determine *reachability*: whether a particular point (usually a link endpoint) of a linkage can reach, that is, coincide with, a given point of the ambient space. Here the configuration that achieves the reaching is considered irrelevant. This is typical of robot arm applications, where, for example, a soldering robot arm must reach the soldering point, but whether the elbow is raised or lowered when it does is of less concern. The final and least specific problem we consider concerns what we will call *locking*: Are every two legal configurations of a linkage connected in the configuration space, or might a linkage be *locked* or "stuck" in one component of the space and thereby isolated from configurations in another component? There are several variations on this theme for different linkage structures, which will be detailed later (in Chapter 6).

The second problem parameter is the answer desired. *Decision problems* seek YES/NO answers: for example, can the arm reach this point? *Path planning problems* request more when the answer is YES: an explicit path through the configuration space that achieves the reconfiguration.

Finally, a third problem parameter is the complexity measure employed. For path planning problems, the combinatorial complexity of the path may be of interest, for example, the number of constant-degree piecewise-algebraic arcs composing the path. But in general, the algorithmic computational complexity is the primary measure: for example, $O(n^p)$, $\Omega(n^q)$, NP-complete, NP-hard, PSPACE-complete, PSPACE-hard, and so on.[4]

These parameters collectively map out a rich terrain to explore, as indicated in Table 1.1. Before embarking on this study, we quickly survey some of the applications that inspired the models.

1.2 APPLICATIONS

1.2.1 Robotics

It is sometime useful to view an articulated robotic manipulator as a chain of links. For example, the robot arm shown in Figure 1.2, developed by Forward Thompson Ltd, is a six-axis (6 degrees of freedom) articulated linkage that is designed to apply adhesive

[2] In the literature, a configuration is sometimes called a placement, confirmation, or realization. We use the term "realization" for a different notion in Chapter 4.
[3] This is also called the *moduli space* of the linkage.
[4] Refer ahead to page 22 for a brief tutorial on complexity classes.

Table 1.1: Classification parameters for 1D linkage problems

Focus	Parameter	Values
Linkage	Graph structure	General, tree, polygon, chain
	Intersection constraints	None, obstacles, simple
	Dimension	2D, 3D, 4D, kD
Problem	Geometric issue	Reconfiguration, reachability, locked
	Answer desired	Decision, path planning
	Complexity measure	Combinatorial, computational bounds

Figure 1.2. A six-axis robot arm. [By permission, Forward Thompson Ltd.]

tape to the edges of plate glass units for protection. The settings of the four joints determine the exact position of this *robot arm*, and so its configuration may be specified by a point in a 4D *configuration space*. Construction of this configuration space for a specific robot arm and analysis of its geometric and combinatorial structure have proven to be a fruitful methodology (e.g., for "workspace clearance") since its introduction by Lozano-Pérez and others in the 1980s (e.g., Lozano-Pérez 1987; Lumelsky 1987).[5] We will make essential use of configuration spaces in Sections 2.1.1 and 5.1.1.2, and in fact throughout this book.

The complexity of the configuration-space approach for robots with a large number of degrees of freedom has led to probabilistic methods (e.g., Barraquand and Latombe 1990), an area which, a decade later, is dovetailing with protein folding, as we will see below.

Another concern of robot designers is *inverse kinematics*: given a desired tool position, compute joint angles which achieve that position. We will touch upon this topic ("reachability") in Section 5.1.1.2. There is currently great interest in the graphics community in inverse kinematics for animating articulated human models (e.g., Zhao and Badler 1994).

1.2.2 Mechanisms

The study of linkages, and more generally, mechanisms, has long been important to engineering. Often the kinematics of mechanisms is of central concern in practical applications (e.g., Hunt 1978), but for our purposes only the geometry of linkages will play a role. A *pantograph* is a typical useful linkage. Its essence is a parallelogram linkage, one of whose joints has its movement duplicated by an attached bar (see Figure 1.3; here the scale factor is 2). The pantograph has been used for centuries to copy and/or

[5] The roots of this idea go back to Lozano-Pérez and Wesley (1979).