6.890: Algorithmic Lower Bounds: Fun With Hardness Proofs Fall 2014

Lecture 3 — September 11, 2014

Prof. Erik Demaine *Scribe: Chelsea Voss, Jonathan Weed*

1 Overview

Last time, we established the hardness of two fundamental problems, (2-)Partition and 3-Partition, and exhibited a bunch of reductions from those problems to other numerical and geometrical ones.

Today, we continue with reductions from 3- and 2-Partition to geometrical problems—we'll also use the fact that the problem of packing n squares into a square without rotations is strongly NP-complete, as we showed last time.

2 Main Section

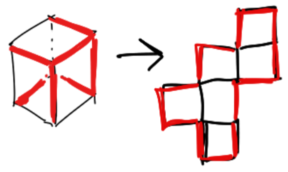
2.1 Edge-Unfolding Polyhedra

We begin with a new problem: Edge-Unfolding Polyhedra.

Problem: Given a polyhedron, is it possible to cut along its edges to unfold it into a connected flat piece with no overlapping sections?

Indeed, it *isn't* always possible [BDD+98][BDE+03], and the above decision problem is strongly NP-complete, even for orthogonal polyhedra (all edges meet at right angles) topologically equivalent to a sphere (no intersections or holes) [AD11].

Figure 1: Unfolding a cube, as an example. We can always cut along a minimum spanning tree of the edges of a polyhedron; the difficulty lies in cutting so that the result lies flat.



2.1.1 Reduction from Square Packing

Very complicated proof! But we'll reduce from Square Packing.

Question: There don't seem to be any numbers here, so in what sense is this problem *strongly* NP-complete?

Answer: There are numbers encoded in the coordinates and sizes of the faces, and in this construction they are all polynomially large.

The intuition is as follows: the infrastructure of the polyhedron will contain a large square hole, with a lot of smaller square faces. Unfolding the polyhedron without overlap will require to fitting those square faces into the large square hole, which is exactly the reduction we seek.

However, in a standard square packing problem we're allowed to move the squares around more or less freely within the larger square. Squares on the faces of a polyhedron are obviously quite constrained in their movements, so how do we construct squares that can move freely?

2.1.2 Atoms

The most important "glue" in the construction is the idea of an *atom*, a gadget that sits on the face of a polyhedron and allows pieces of the unfolding to mover arbitrarily. Atoms are "bumpy squares" that can go left/right on the surface of the polyhedron and move a *different* direction when they're unfolded. Connecting the squares with atoms solves the problem of moving each of the squares without constraint—see the paper [AD11] for more details.

2.2 Snake Cube (Cubra)

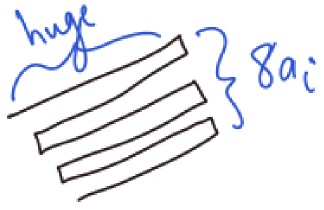
The Snake Cube (or Cubra) is a set of small cubes connected by elastic, which constrains the movement of the cubes. There are straight sections, which are constrained to lie in a line, and bends, which can move 90° in any direction. The goal is to to assemble the string of cubes into a single larger cube. Deciding whether a Snake Cube puzzle is solvable is NP-complete [ADD+13].

2.2.1 Reduction from 3-Partition

Once again, we have two (main) issues in reducing an instance of 3-Partition: the first is finding gadgets to represent the a_i, and the second is finding glue that will allow the a_i to move in an unconstrained way.

The a_i will be represented as very long runs of straight cubes that double back on themselves $8a_i$ times.

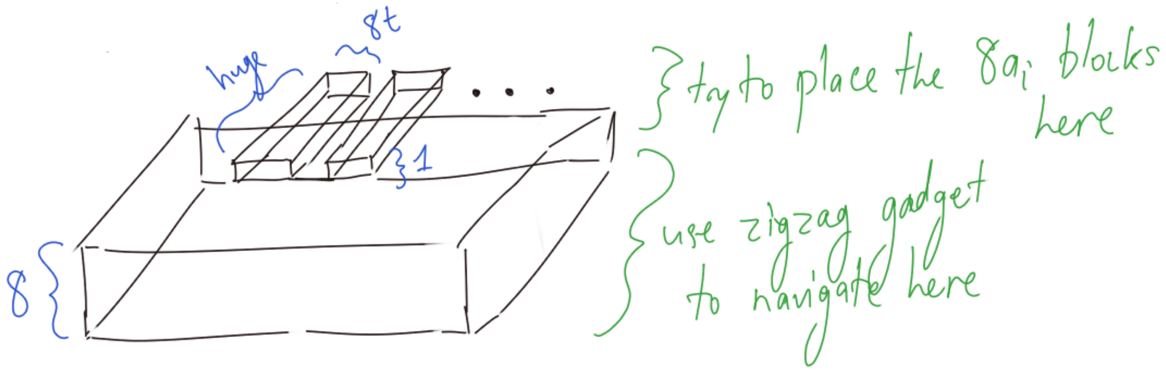
Figure 2: The a_i gadget.



The runs of cubes are so long that they can only fit in one place: long spokes in the construction corresponding to subsets of size exactly t. (The turns are very tight so we can't "break up" an

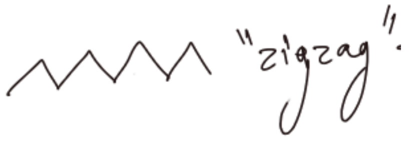
a_i—this is going to be a big theme in 3-Partition reductions.)

Figure 3: The reduction from 3-Partition requires that the snake be folded to fit into the following space. Long spokes contain the a_is, and the larger block is there to allow movement from spoke to spoke.


8t
huge
1
8
try to place the $8a_i$ blocks here
use zigzag gadget to navigate here

The glue in this case is the "zigzag"—a chain consisting entirely of bends—which is a sort of universal shape.

Figure 4: The "zigzag" gadget.



False statement: Zigzags can make any polycube shape (any 3-dimensional agglomeration of cubes along faces).

This is certainly false (for instance, a zigzag can't form a straight segment). But things get better if you scale any polycube up by a factor of 2 in each dimension (or, equivalently, refine each cube into a $2 \times 2 \times 2$ cube composed of 8 smaller cubes).

Truer statement: a $2 \times 2 \times 2$ refinement of zigzag can make any polycube shape.

This also isn't quite true, because while we can't make a self-intersecting shape—we're limited to a Hamiltonian shape (one path with no collisions). But, by a theorem in computational geometry, we can refine *again* to remove this obstruction.

Truest statement: a $4 \times 4 \times 4$ refinement of zigzag can make any shape.

Scaling up by a factor of 2 turns any path into a Hamiltonian path, and we're done—a zigzag shape can make any polycube shape.

We should note that there is a parity consideration. (This'll be really important for other constructions in this class.) Each link in our chain moves from a cube of one parity to a cube of the other parity. We always *enter* and *exit* our $4 \times 4 \times 4$ cubes through two cubelets of opposite parity. The $4 \times 4 \times 4$ refinement of zigzag can make any shape—but paths always satisfy the parity condition.

2.2.2 Oh, one more thing...

The problem was to arrange our Snake Cube into one larger cube, but the shape we were packing into wasn't cubical!

The solution is to use very long runs of straight blocks, which have no choice but to lie flush with the edges of the large cube. By creatively arranging enough of these, we can "carve out" the infrastructure we need to encode the rest of the 3-Partition structure, above.

Remark: There's a lot of algorithmic thinking here—designing the right algorithms can provide you with powerful gadgets to build the right structures for your reduction.

2.3 Disk Packing

Problem : Given a square and a collection of disks, can you fit the disks into the square ((o be more precise, so that their centers are in the square)?

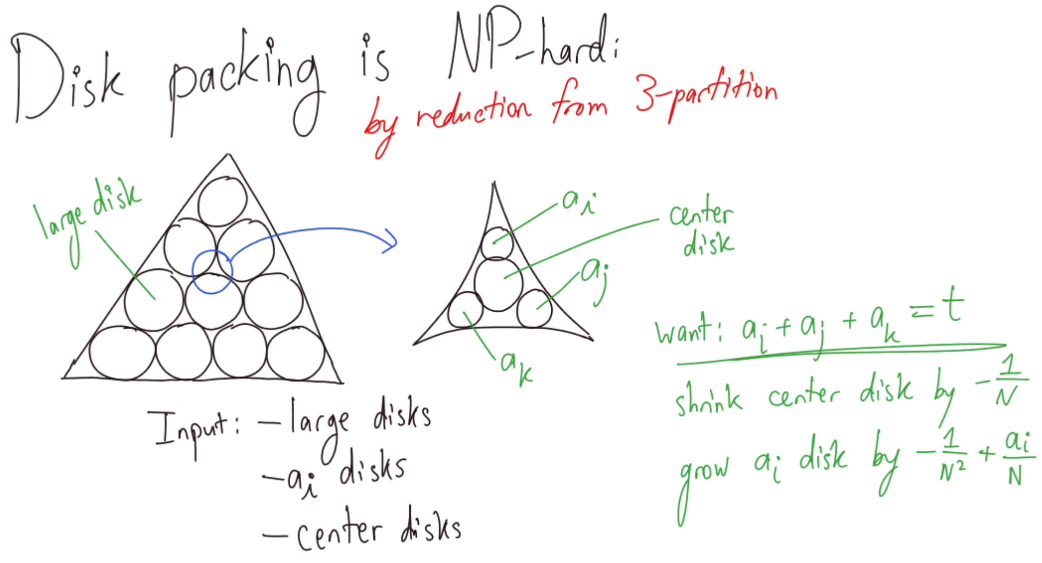
This is a real life problem, in the sense that a disk packing algorithm can design origami structures by the tree method. People have thought about this problem a lot, but it was only recently proven NP-hard [DFL10].

2.3.1 Reduction from 3-Partition

Begin by thinking about packing an equilateral triangle rather than a square. We set up a triangular "grid" of large congruent disks, and then build 3-Partition gadgets in the space between them, each of which will represent one of the subsets in our partition.

We reduce the elements a_i to circles with diameters based on the size of a_i in such a way that three circles corresponding to a_i, a_j, and a_k fit if and only if $a_i + a_j + a_k \leq t$.

Figure 5: Disk packing in a triangle.

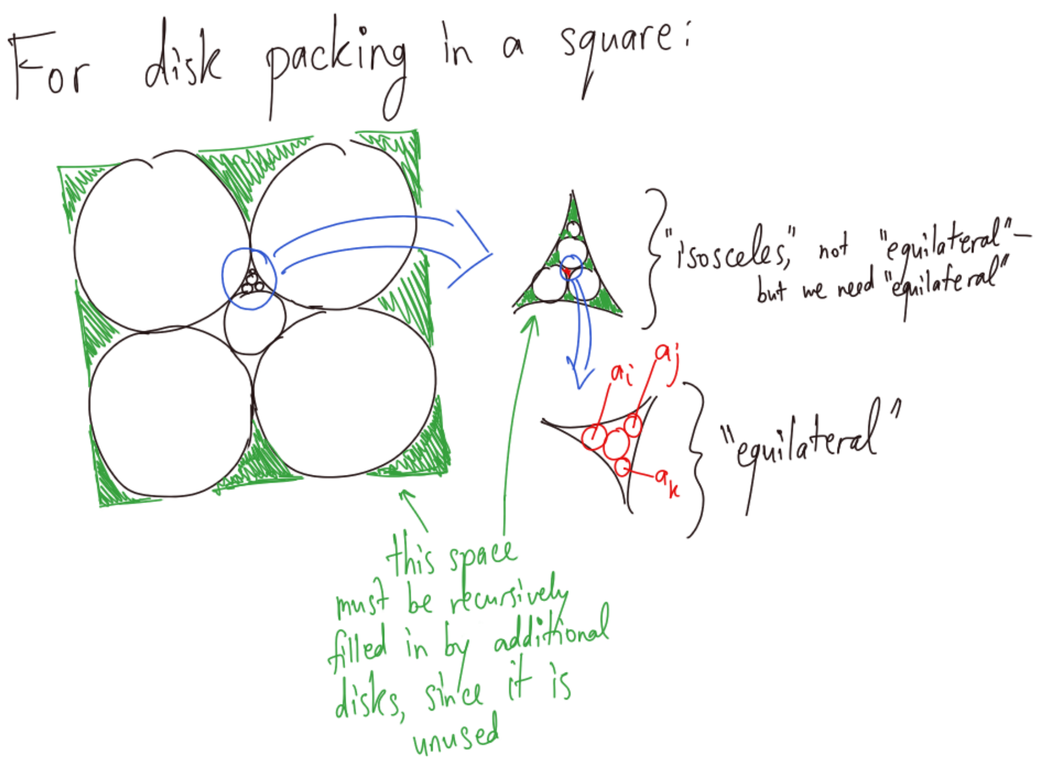

Disk packing is NP-hard:
by reduction from 3-partition
large disk
a_i
center disk
a_j
a_k
want: $a_i + a_j + a_k = t$
shrink center disk by $-\frac{1}{N}$
grow a_i disk by $-\frac{1}{N^2} + \frac{a_i}{N}$
Input: – large disks
– a_i disks
– center disks

Working this out requires some subtlety because circles aren't straight (they're circles, after all...), so sums don't map so clearly to the dimensions of circles. But they're like segments up to first order, so we use Taylor expansions to calculate the proper radii. In particular, we shrink each circle by an amount $-1/n^2 + a_i/n$, where the $1/n^2$ term cancels higher-order terms in the expansion.

Comment: The point of showing you all this is to give you lots of different ways to represent numbers. The more ways you have of representing numbers, the easier it will be to find reductions from 3-Partition to geometric (and other) settings.

Finally, we have the issue of packing everything into a square. The general idea is to fill up the grid with lots of circles of decreasing sizes in $O(\log n)$ steps, which constrains the gadgets to sit in the right place. By considering circles in decreasing size, we can be convinced at every stage that infrastructure circles are constrained.

Figure 6: How to adapt disk packing to a square instead of triangle.


For disk packing in a square:
"isosceles," not "equilateral"—
but we need "equilateral"
a_i
a_j
a_k
"equilateral"
this space must be recursively filled in by additional disks, since it is unused

2.4 Clickomania

Clickomania is a computer game, and was Erik's first (!) hardness result [BDD$^+$02]. The goal is to clear blocks by clicking contiguous groups of n blocks ($n > 1$) of the same color, which is thereby destroyed. Blocks fall column by column to fill up empty space, and empty columns disappear from the board. Solving Clickomania with only one column is in P, by reduction to a context-free grammar. However, solving the Clickomania problem with 2 columns and 5 colors is NP-Hard (reduction to 3-Partition), as is Clickomaina problem with 5 columns and 3 colors (reduction to 3-SAT).

2.4.1 Reduction from 3-Partition

In the 2 column/5 color case, we set it up so that the right column encodes the a_i, and the left column encodes the desired sum t of each partition.

The left column is mostly a symmetric checkerboard pattern interspersed with red squares; the only way to clear the entire column is to clear all the red squares (and thereby solve 3-Partition). Clicking on blocks in the right column corresponds to choosing elements for one subset A_i; when that subset reaches the desired sum, two red blocks in the left and right columns line up exactly and we can clear both. There are a lot of small numerical details to check, but otherwise the construction is straightforward.

2.5 Tetris

Tetris is a computer game played on a rectangular board with falling tetronminos (4 unit squares joined along edges). Each block can be rotated as it falls from the sky, and filled rows disappear. You lose if your pile of blocks ever reaches the top edge of the board (the "sky").

Problem: Given an initial board and the entire sequence of future pieces, can you survive?

Note that this is a problem with perfect information: we know everything about the current and future states of game. This problem is nevertheless is NP-complete, a result from early in Erik's career [BDH$^+$04].

It is also NP-hard to approximate the number of lines or blocks until death up to a factor of $n^{1-\epsilon}$.

2.5.1 Reduction from 3-Partition

(We have to reduce from a strongly NP-complete problem because the proof will require encoding the a_i in unary.)

The starting board configuration contains gadgets called "buckets", each $\Theta(t)$ call. Filling these buckets corresponds to choosing a_i to fill your subsets. Once these buckets are filled, a single T-block can unlock an empty column to the right of the playing field, which can be filled by straight pieces. Without unlocking this column, it's impossible to survive.

The a_i's correspond to a sequence of tetrominoes which fit perfectly in buckets. A key problem for the proof is the idea of splitting: since each element a_i is a series of pieces, there's the danger that the player will split the a_i's into two or more buckets, which ruins the correspondence between Tetris and 3-Partition. The solution to this problem is to introduce a priming piece between each a_i. The priming piece "unlocks" a bucket, and trying to split an a_i won't properly prime the bucket, and won't lead to a solution.

2.5.2 Approximation

By building this "lock" structure atop a large, empty space, we can ensure that opening the lock will increase the number of lines before dying by an exponential factor. This means that even approximating the number of lines a player can survive to within $n^{1-\epsilon}$ is NP-hard.

2.5.3 Open Problems

Figuring out how hard Tetris is in any of the following situations is still open:

- Tetris with an initially empty board
- Tetris with $O(1)$ rows or columns
- Tetris with restricted piece sets (For instance, Tetris with only straight pieces is trivially easy. What subset of pieces is required to make Tetris hard?)

- Tetris without last-minute slides (where ever piece needs to be dropped from the sky)
- Online Tetris (Tetris where you don't know the future sequence of moves)
- 2-Player Tetris (which might be PSPACE-complete)

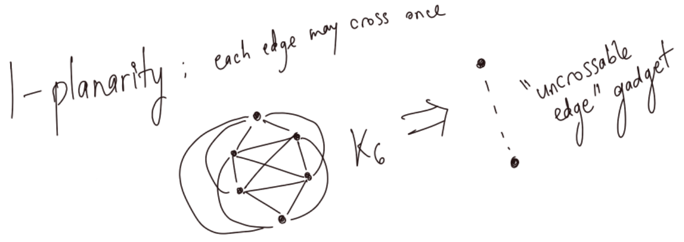
2.6 1-Planar Graphs

1-Planar graphs are "almost" planar graphs: each edge can cross at most one other edge. Deciding whether a graph is 1-planar is NP-complete [GB07].

2.6.1 Reduction from 3-Partition

The key idea in this construction is the creation of an uncrossable edge. An uncrossable edge is a subgraph (in fact, K_6) with the property that any edge passing through it must cross an edge that already has a crossing, so crossing this subgraph isn't allowed. Once we've created an uncrossable edge, we can use it as a black box to create graphs that only allow crossings in particular places.

Figure 7: The "uncrossable edge" gadget.


1-planarity : each edge may cross once
$K_6 \Rightarrow$
"uncrossable edge" gadget

Lesson: It's often useful to create black box gadgets that you use as your notation—once you've established them, the rest of the construction becomes a lot easier.

We build two wheels out of uncrossable edges, one corresponding to the set A and one corresponding subsets of A. Using uncrossable edges, we can make each subset separate and force each a_i to belong to only one subset. Lots of cases to check, but uncrossable edges make the picture a lot clearer.

2.7 Ivan's Hinge

Ivan's Hinges are hinged loops of colored triangles which can be folded into colored shapes in the plane. Deciding whether a puzzle can be folded into a certain shape is NP-complete, by another reduction from 3-Partition [ZDD+14].

This problem also involves finding universal shapes that can form any other shape (in this case, called the GeoLoop), and the a_i are represented as colored blocks in the final construction. Choosing subsets corresponds to filling up contiguous regions of colored blocks, with the rest of the loop dedicated to allowing the a_i to move around freely.

2.8 Carpenter's Rule

An "annoying" Carpenter's Rule is a foldable ruler of unit width, where each straight section can double back on the next. The goal is to fit the ruler into a long straight box of a specified size. This problem is *weakly* NP-hard [HJW85], and, like 2-Partition, admits a pseudopolynomial algorithm.

2.8.1 Reduction from Partition

Choosing which of the two subsets a_i should fall into corresponds to choosing whether a segment of length a_i goes left or right. If the two subsets are of equal size, then our ruler will start and end at exactly the same point.

Adding two long segments and two slightly shorter segments at each end, we can reduce from the problem of finding two equal subsets to the problem of fitting the ruler exactly into the box.

2.9 Map Folding

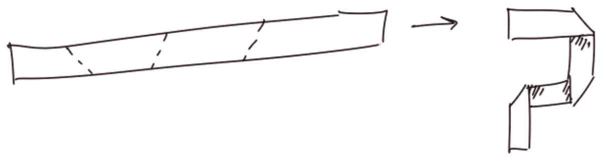
The problem is to decide whether a map and a given pattern of creases is foldable. This problem *weakly* NP-hard [ABD$^+$04] for orthogonal paper and orthogonal creases or for rectangular paper and orthogonal and diagonal folds. It is not known whether this problem is strongly NP-hard or if there exists a pseudopolynomial algorithm.

2.9.1 Reduction from Partition

The reduction is similar to the reduction for Carpenter's Rule. There are horizontal folds corresponding to the a_i and two vertical folds corresponding to the division between subsets. We can fold some subset of the a_i first, then fold the vertical folds, and then fold the remaining a_i. The creases will match up perfectly if and only if the two subsets have the same size.

This construction uses "orthogonal paper," which doesn't much resemble a traditional map. But adding diagonal creases to a rectangular piece of paper can force it to fold into an orthogonal shape, which can then be used to encode an instance of partition using our reduction for orthogonal paper.

Figure 8: How to form "orthogonal paper" from a long rectangular paper using diagonal creases.



References

[ABD$^+$04] Esther M Arkin, Michael A Bender, Erik D Demaine, Martin L Demaine, Joseph SB Mitchell, Saurabh Sethia, and Steven S Skiena. When can you fold a map? *Computational Geometry*, 29(1):23–46, 2004.

[AD11] Zachary Abel and Erik D Demaine. Edge-unfolding orthogonal polyhedra is strongly np-complete. In *CCCG*, 2011.

[ADD$^+$13] Zachary Abel, Erik D Demaine, Martin L Demaine, Sarah Eisenstat, Jayson Lynch, and Tao B Schardl. Finding a hamiltonian path in a cube with specified turns is hard. *Information and Media Technologies*, 8(3):685–694, 2013.

[BDD$^+$98] Therese C Biedl, Erik D Demaine, Martin L Demaine, Anna Lubiw, Mark H Overmars, Joseph O'Rourke, Steve Robbins, and Sue Whitesides. Unfolding some classes of orthogonal polyhedra. In *CCCG*, 1998.

[BDD$^+$02] Therese C Biedl, Erik D Demaine, Martin L Demaine, Rudolf Fleischer, Lars Jacobsen, and J Ian Munro. The complexity of clickomania. *More games of no chance*, 42:389, 2002.

[BDE$^+$03] Marshall Bern, Erik D Demaine, David Eppstein, Eric Kuo, Andrea Mantler, and Jack Snoeyink. Ununfoldable polyhedra with convex faces. *Computational Geometry*, 24(2):51–62, 2003.

[BDH$^+$04] Ron Breukelaar, Erik D Demaine, Susan Hohenberger, Hendrik Jan Hoogeboom, Walter A Kosters, and David Liben-Nowell. Tetris is hard, even to approximate. *International Journal of Computational Geometry & Applications*, 14(01n02):41–68, 2004.

[DFL10] Erik D Demaine, Sándor P Fekete, and Robert J Lang. Circle packing for origami design is hard. *arXiv preprint arXiv:1008.1224*, 2010.

[GB07] Alexander Grigoriev and Hans L Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007.

[HJW85] John Hopcroft, Deborah Joseph, and Sue Whitesides. On the movement of robot arms in 2-dimensional bounded regions. *SIAM Journal on Computing*, 14(2):315–333, 1985.

[ZDD$^+$14] Zachary Abel, Erik D Demaine, Martin L Demaine, Takashi Horiyama, and Ryuhei Uehara. Computational complexity of piano-hinged dissections. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 97(6):1206–1212, 2014.