PROFESSOR: Today, we continue-- we're going to do two things today. One is protein folding and the other is interlocked chains. So protein folding, we're continuing on where we left off last time, which was essentially looking at the mechanics, the mechanical models of protein folding, which was fixed angle chains. But basically ignoring all the forces that go into actual protein folding.

Today, we're going to look at some of the very theoretical work on what forces cause proteins to fold and what you can say about algorithms and complexities of those problems. So obviously, a lot of people work on protein folding. On the practical side, people look a lot at defining energy functions that you minimize in order to find what we believe would be the right protein folding.

There's a lot of energy functions around, and the key energies that-- forces people look at are torsion angles potentials, van der Waal interactions, hydrogen bonds, hydrophobicity, secondary structure propensity, and paralyze specific interactions. And then they take some weighted combination of all those things and try to find one that matches up with what we see in real life.

The things we see in real life, we get by crystallizing proteins, which can take a really long time, like years of research. Then you get one 3D image. And that's what protein data bank collects is all those 3D images. People try to train against that data, but I say, we really don't know what the right objective is for real protein folding.

Given that, because I'm a theoretician, we're going to look at a super simple model that just models one of those features that I mentioned, which is hydrophobicity. Hydrophobia is fear of water. And remember we have this back on the protein, the various amino acids hanging off, some of those amino acids are hydrophobic, meaning they like to hide from water.

Now usually, your proteins live in a big bath of fluid, mostly water, and so they're trying to hide from the surroundings, try to cluster on the inside of the folded shape, whereas other hydrophilic amino acids that like water, they tend to go around the outside. You can't always do this, but you try to minimize the amount to which the hydrophobic guys are on the outside and maximize the hydrophilic guys are on the outside.

In the HP model, there are hydrophobic and hydrophilic guys. And actually, we'll call h the hydrophobic and P the hydrophilic. It's very confusing because both of them have h and p in the acronym, but p also apparently stands for polar, polar opposite of the hydrophobic guys. Anyway, we just have these two kinds of nodes, and we just try to model that one feature, plus the bonds that link together, the backbone of the chain.

And so we're going to model a chain as just a sequence of nodes. We're going to fold that chain on a lattice. Something like this. Is it five? Yeah. And different of these nodes could be marked h or p. And we usually could use two colors to denote that. In this case, I made this guy h, this guy h, this guy h.

The score of a fold-- so this is a folding. We're always going to fold on the square lattice today although people have looked at other lattices, in particular 3D cubic lattice and a little bit on the triangular lattice. I'll mention some reasons why the square lattice is a little bit artificial, but in general, conjectures, it doesn't really matter which lattice you fold on.

But you really want to follow lattices because we're going to define the score in terms of how many adjacent H nodes there are on the lattice. Score is the number of call them H-H bonds. The intuition is if you have two hydrophobic nodes that are next to each other that means they're not next to the boundary, and so they're happier.

There's a more formal argument of that, but this is an easy definition of score. You want to maximize the number of H-H nodes that are adjacent on the lattice. Usually, there are some that are adjacent along the chain. Those we don't care about because they will always be adjacent. You have to fold this thing so that consecutive nodes in the chain appear adjacent in the lattice. So we usually ignore what I would call H-H edges which are four, so you can count them or not. It doesn't really matter.

We want to maximize the number of these guys. For this chain, that's probably the maximum you can get. In general, a typical node has two edge neighbors, and it has two other neighbors. And so at most, every guy has-- every H node has two bonds next to it, which means on average at most one bond per H node because we're double counting. So that's the model.

The optimal folding has the maximum score. And I'm going to talk about the theory of this model in a moment. Some practice you might say is this picture. This is heuristic or local optimization of maximizing. Here, red nodes are H nodes that will be throughout the slides. Red nodes are H nodes. Blue nodes are P nodes. That's a universal standard.

You can see, in this case conveniently-- it may look like some random pattern of reds and blues, but you end up with all the blues pretty much covering the outside, and so the reds cluster together, and there's lots of H-H bonds. They are not drawn here, but-- So that's nice. And this looks like a typical protein in that it globs together. But of course, it's not capturing every feature of protein folding, just trying to capture one of them.

But pretending this is the whole story, what can you say about optimal folding in the HP model? Well sadly, it's NP-hard. So to find the actual optimal folding of a given string of red and blue nodes is NP-hard. It's NP-hard in the 3D cubic lattice. That was by two MIT professors, Bonnie Berger and Tom Leighton

And then it was proved NP-hard hard in two dimensions by whole bunch of people, I guess, from Berkeley. So that's bad news. And to me, that's a sign that nature probably does not find the optimal folding. There are a couple possibilities here. Maybe they're extra forces beyond hydrophobicity that make it easier to find the optimal solution or maybe nature's just finding a local optimum, not a global optimum. Or maybe the proteins that exist in real life that have been evolved are designed so that local optima equal global optima. That's somewhat related.

Somehow nature gets around this NP-hardness, I would think, because I believe nature's a computer. Maybe not everyone does. But it should be bounded by polynomial time computation. So unless P equals NP-- that's another possibility I guess. It's unlikely that, to me, nature is doing this. But maybe it's finding local optimum. That seems to do pretty well in this particular example.

AUDIENCE: Hey Erik?

PROFESSOR: Yeah.?

AUDIENCE: Are there approximation algorithms?

PROFESSOR: Good question. Are there approximation algorithms, that's exactly where I wanted to go. There are lots of constant factor approximations, and I will show you one of them. So I'm not going to show you these hardness proofs because they're a bit complicated. I don't have free diagrams of them. But the constant factor approximations, I do. Where are we going? Here.

And so, in particular, the best approximation that's known is a 1/3 approximation. This is by MIT Ph.D. student [INAUDIBLE], and from several years ago-- eight years ago. I believe it's still the best. It's pretty simple, but quite elegant, looks like protein folding. So this is going to be within a factor of three of the best you could hope for. We write 1/3 because we're always getting fewer bonds than you might want in a maximization problem.

Some people call this three approximation. I call it one of the other depending on the day the week. So how could we achieve 1/3 approximation? Well, before I get to this image, let me tell you. In general, when-- we haven't talked too much about approximation algorithms in this class, but a typical technique in approximation algorithms is you get some bound on what the optimal might be.

I already told you there's at most one bond for each H node. And that is indeed-- that's roughly what we'll say. Just because of the degree argument, every guy has bond degree at most two. So that means, at most, one edge per H node. But we can be a little more precise on the square grid because there are two types of nodes.

If you checkerboard color the square grid, there's even nodes and odd nodes. So in fact, we claim that the optimal solution, which we call OPT, is always at most twice the min of the number of even H's and the number of odd H's. This is slightly smaller than the number of H's. If a number of H's and not H's are equal, then this will just be the number of H's. But if one is smaller than the other, this will be a little bit smaller or it could be a lot smaller. They're saying, no even H's.

The point is if you have an even parity H, and it somehow has a bond with some other H, well obviously, that h has different parity from this one. So only even and odd H's could possibly bond together. And then this is pretty obvious. Let's see if I can formalize it. So you have every bond defines and even and an odd. Every even and odd can only get hit up to twice.

If there's more evens and odds, it's not going to help you. In the best case, you pair them up or alternate even odd, even odd, even odd. So you'll never be able to use more evens than odds, maybe one more, not really. And I think the best case is you do a cycle of even odd, even odd, even odd. Then, there's an even number of them-- each, an equal number of each.

And so you throw away the excess. You end up with a min of the two. But you get a factor of two because every even guy has two incident bonds in the best case, and then you're not double counting because you count this even guy, maybe you count this even guy, and they have disjoint bonds. So it's twice the number of evens or number of odds, whichever is smaller.

So this is useful. It's just what is the best case you could hope for OPT? Now, what this 1/3 approximation will get is at least 1/3 this amount. If you know you get at least 1/3 this amount, you know you get at least 1/3 the optimal because this is an upper bound on the optimal. So that's with the algorithm will do.

And so the general idea of the algorithm is to proceed in stages. At a typical stage--

So you pick some edge. I don't think it'll matter too much. Pick some edge of the chain, and then you start working your way down. And general idea is you can make big loops off to the sides so that when you get back to the center, these are H nodes. So you get a bond there. That's the general idea. With that basic idea, you probably get about 1/4 of this amount. But we want to get 1/3, which is a little better.

I won't try to analyze that formally. But there's a lot of flexibility here when we-- how long we make this branch. In order to make this work out, what I like is that the left branch has a lot of-- pick one-- a lot of even H's, and the right branch has a lot of odd H's, or vice versa because then we'll be able to make lots of the even guys match up with the odd guys.

Here, we're fighting the parity. And parity is one of the reasons why I think the square grid is a little bit artificial. Let's say with the triangular grid, you wouldn't have that. There aren't just two classes. But here, we can get around parity by finding a break point, this edge, so that the number of even guys in the left is about the number of odd guys on the right.

I won't prove that, but it's just an intermediate value argument. See, you get that as you walk counterclockwise along the chain, you have more even guys, and walk clockwise around the chain, you get more odd guys. Once you have that set up, you fall into four cases.

So we're supposing at the top-- let me get to a case, that I can reach. Here's a typical case. Suppose you just succeeded in making two H nodes join together. Maybe that was your first edge, whatever. So it might not be H-H. But now, we're proceeding down, and we want to align a bunch of H nodes. And we're going to do better than the picture I drew over here, which was just getting two H nodes to come together.

I actually want to get four H nodes to come together with three bonds. That's what I will always achieve, at least that. Is that right? Actually, no, this is the good case. I won't always achieve quite that well, quite that good. But here's the idea. Suppose there's some reasonably large distance between this H node and the next one.

I'm going to go down two steps and then go around however long it takes so that when I come back, I get to an H node, and same on the left side. So these two chains may have different lengths depending on when the next H node is. Now, when I say the next H node, I actually mean the next odd H node on this side and the next even H node on this side so that they will line up.

This is assuming that the distance from this H node to the next one is greater than 1, by parity that means it's at least three. So if it's three, it's going to be one, two, three. And that's it. And so go straight down. If it's more than three, you just divide that distance in half, and you go back and forth. And this sets the parity right.

So that's the set up. Now, we have these two H nodes. Now, when are the next H nodes? Well, there's at least one blue node in between because of parity because we're looking again at odd H nodes on the right side, even H nodes on the left side. So maybe it's just one blue node, and we would go like this, and here we would go like that.

Or if it's more than one blue node, we just lay them out like this. Goes to the appropriate distance, so that when we come back, we get a red. So we have two red H nodes here. I want to make two red H nodes here no matter what. If it's just gap of one, we would go straight, make a corner, otherwise, we do these big loops.

And the result is I get these four H nodes in this pattern, just three bonds for four H nodes. And before we analyze what exactly that means, let me just tell you about the other cases. So this is the case when there is more than one blue node in between two H nodes. The other rows and columns are when the left side or the right side has just one blue node from that red guy.

So let's go to this one for example. Here the left side has only one blue node to the next red guy. The right side has at least three. So then we'd go one, two, three, and we get there. If it's a more than three, then we go and we make that big loop. Now we have these two read nodes. My goal now is to put a red node here.

If this chain only has one blue node, I would make a corner. Otherwise, I'd loop around, and then I go onto the next step. Here, I only got two bonds for three nodes, so it's actually a little worse in ratio. In the case where they're both length one, I just go straight down, make those two H nodes, and then I use the same trick as over here to get that nice zig-zag pattern. And this case is symmetric to this one, just flipped.

So in all cases, I didn't skip anything. I did all of the even H nodes on the right side. I did all of the odd nodes, H nodes, on the left side. So I did skip something. I skipped all of the odd guys on the left side and all the even guys on the right side. So I skipped half of the H nodes. So I get at least two bonds-- I'm calling them H-H bonds-- for every three interesting H nodes.

By interesting, I just mean even on the left, odd on the right, which is really for every six H nodes. And so we get a factor of 2/6, also known as 1/3. Really here I mean 6 even H nodes. Let's say. We used the same number of even and odds. And we had-- Is that right? Sorry. No, I guess I actually made H nodes here. I always get a little confused here because there's a factor of two, a factor of two here.

But in the end we get 1/3 bound. So that's the approximation algorithm, and that's the best node. It looks pretty nice. It's like beta sheets, if you've seen proteins folding. it depends, of course, on the spacing of the H nodes, but you get-- in particular, you get all the H nodes on the inside pretty much.

I guess here is one exposed end. And here there's one exposed end. These two cases, we're doing especially well. Maybe in real proteins, the gaps are always more than one. I don't actually know. I think in reality there's different levels of hydrophobicity, so that's maybe not so well defined. But any questions?

AUDIENCE: Is the exposed end considered the [INAUDIBLE] above the dashed line or is it--

PROFESSOR: Yeah. When I say exposed end, I mean here. There's an adjacency to the outside water. Now of course, there might actually be nodes here. Then, that would not be an exposed end. But if this just went straight up, which it might in this picture, then that would be an exposed end. I guess that might also be. No, probably not.

Actually, interesting. If these two guys are red, there's always a blue node to the left and right of them in this construction, so actually, I think that would not be exposed. That's cool. So actually, I think there are no exposed ends in this folding, except maybe at the very top and bottom. Except, we're only considering half of the H nodes, so there's the other half we have no idea about.

So hopefully, nature chose the parity right. Or, if you think about this is in the triangular case, which I don't think anyone's tried to adapt this algorithm to the triangular grid, maybe you can get no exposed node. So that would be pretty neat. Of course reality, it probably does not fold on a lattice, but it's an approximation.

Now, one of the observed features in protein folding is they tend to fold to the same shape. It's actually a hard thing to measure, but that's the general consensus, that at least most proteins in a non diseased organism fold always to the same shape, at least in the same environment. So if you want to model that in the HP model, and you pretend that, really, we are finding optimal solutions-- because it's a little hard to define what locally optimal means. But that would be an interesting direction.

It would be nice to find proteins where locally optimal equals globally optimal, but lacking definitions of those terms, no one has tried to prove that. It could pick a good direction. But if you believe in optimality, then at least you hope that the optimal are unique. So you take a typical example of the optimal, like if they're all blue, then every folding is optimal. So there's exponentially many of them.

But if you have any hope of getting unique folding, you'd really like unique optimal foldings. The theorem is these exist. And the 2D squared grid, they exist for all even n, for closed chains-- mostly, we care about open chains, but for closed chains, you could prove these exist for all even n-- and for open chains, they exist for all doubly even n.

This is a theorem by a bunch of people. Oswin Aicholzer, David Bremner, me, Hank Meijer, Vera Sacristan, and Michael Soss, we talked about last time. So here they are for closed chains. It's a pretty simple example, basically alternating red blue, but at the end, you have two blue nodes to make the corner. That is not the optimal folding. The optimal folding is this.

So there's only two exposed ends. There's one down here, and one on the top. So there's no surprises that it's optimal, a little less obvious that it's uniquely optimal. But there it is. And this works for any even n. In general, it depends a little bit on your the n mod four, so that's why we are drawing two pictures here, to check that it works for both congruences mod four.

It has to be even for closed chain and must be even, so this is really for all m that there's closed chains, there are unique-- there are examples of HP chains that have unique optimal foldings. Here, it's just to do you end going horizontal or n going vertical? Not a big difference. I have a proof by picture that this is the optimal folding.

So remember there are two-- we talk about H-H edges and H-H bonds. These are what we call P-P edges. And I remember when writing this paper, one of the co-authors, I won't say which one, saying call me juvenile, but I don't want to have a paper that talks about P-P edges. And I convinced him to keep it.

So we have-- it's really only obvious when you say it out loud, which sadly I have to do here. But you have two of those edges, all blue edges, we might also call them. And they really should be on the boundary because you have to have exposed ends on the boundary, on the bounding box is what I mean.

So if you put too many red guys on the bounding box, you're going to lose lots of bonds. So those two blue guys should be on the boundary. Some two edges have to be on the boundary, so it's best to have the two blue ones. So we could show you in optimal solutions, that is the case. Now, you decompose into two chains that connect the two ends.

There's connecting this end to the left and connecting this end up and around to the other blue node at the end. And if you check all of the red notes on this side have even parity, let's say, and all the red nodes on this die have odd parity because it alternates. So there's no way that you could have bonds within one of those chains.

That's not drawn here.

The other thing is because these two blue edges are on the boundary, you cannot have a bond on the outside because there's no hope for that. You can only have bonds on the inside of that closed loop. Here, we're using that it's a closed loop. So you have bonds on the inside, and then you argue, well, because it's bipartite graph and because it's planar, you can't have any crossings in these bonds because you're embedding this thing in the plane, really you basically have to alternate.

There's actually two ways you could try to alternate. You start on this side or if you start on this side. You try to realize them geometrically. When you do this alternation, you decompose your thing into squares. Only in one case does it work, and you get this picture. And then once you have the squares, there's a unique way to glue them together. So that's how the proof goes. It's pretty easy once you get the right limits in place for closed chains.

Let me show you what happens for open chains. Take the same example for open chains, and it's almost unique. In the double even case, like 16, it is unique. In the only singlely even, so two mod four case, like 18, you have this issue. So the way we set up is we do not have the blue blue at the end to turn around. That is the best we know how to do.

And there are these two-foldings, how you might achieve it. So here, we wrap around and get a little triad of blues here. Because we have a red n point, we can actually get three bonds into it. Or, we can do the intended folding. Now, it's a pretty small change, so it's approximately unique, but that's the best we know for open chains, only singly even length.

Or for odd length, we also don't know anything that's optimal. Of course there are things that are almost optimal. You just add a little end. Won't make a big difference, but interesting open problem. Also open, whether any of these unique results-- uniqueness results extend to other grids, like triangular grid. That would be nice because here we're really exploiting parity, and that's kind of cheating. Certainly, parity is not such a big deal in real life, but I don't know. Could be.

The approximation algorithm's generalized to other grids, not necessarily this one, but some constant factor approximation have been obtained. Uniqueness is open.

The really cool open question here, I think, is protein design. HP model is simple enough we can think about it. And while folding-- finding the optimal folding of a given protein is NP-hard, if I get to design the protein, and say, well, I'd really like to fold into-- I don't know. The letter m-- how do I make a protein fold into the letter m, at least approximately?

So ideally, it should still have a unique optimal folding, so we generalize this result, but it can match any target shape. Here, we're matching a diagonal line up to some constant factor resolution. Can we match any shape up to some constant factor resolution? I think the answer should be yes. And then you should be able to solve that polynomial time.

In the same way that origami design is a lot easier than origami foldability, here, I think protein design should be a lot easier than protein folding. But no one has worked on that. I think it's a really cool problem. To bad I'm mentioning it so late in the semester, but we can work on it in the problem session. So ends my coverage of protein folding.

Obviously, there's tons of work I'm not covering, but I'm focusing on the very algorithmic stuff. Any questions before we go? Yeah?

AUDIENCE: In these models you have, are these supposed to equal number of P and H, can you get in situations where you're way overbeared with P or H, and then it's going to be harder to find an optimal folding. Or, is it not realistic because these are such-- these are small sets of the entire protein?

PROFESSOR: Yeah. So the question is whether-- Does it matter the ratio between the number of H nodes and P nodes. I think the easy-- A sure thing is I don't know. My guess is, in reality, the number of H and P's are within a constant factor of each other, for some constant. I don't think the constant matters too much. Although if there's an overwhelming majority of H's or P's, the problem might become easier.

My guess also is in the NP-hardness proof although it's been a while since I read it. There's probably a constant factor again. But maybe you could show if there's-- for example, if there's a constant number of h nodes, I'll bet you can solve it in polynomial time. That's maybe-- That's a new open problem. I should write it down. Good question.

AUDIENCE: If there is a wrong factor between them, wouldn't we see that at the model because it just wouldn't work?

PROFESSOR: No, this model makes sense for any number of reds and H's and P's, but uniqueness is probably hard to obtain if you have a small number of H's or if you have a huge number of H's. If you don't have enough--

AUDIENCE: What if you have H's on the order of the square of the P's, then there are at least two H's?

PROFESSOR: All right.

AUDIENCE: Kind of makes sense, right?

PROFESSOR: Yeah, actually we have a theorem along those lines, which is not published so that's why I'm not talking about it. You can achieve the number of H's as square the number of P's. I imagine that's the limit-- and still get uniqueness. I imagine that's the limit. We really ought to write that result that it's from seven years ago, so unlikely at this point. Other questions?

So we go onto interlocked 3D chains, which is related in that it's in 3D and its chains. So related to the mechanical protein folding we did last class, except here, mostly, I'm going to think about universal joints again, so I can do whatever I want. I don't have to hold the angles fixed. At the end, I'll mention how things change a little bit for when you have fixed angle chains. But universals more fun.

So the motivation here is we have our good friend the knitting needles example, or maybe our mortal enemy, depending on how you like to think of that. One, two, three, four, five. And remember-- never mind the edge lengths. I'm not going to worry about edge links here. With five bars, you can lock the chain.

So an interesting question is what if I had less than five chains, say length 4 chains, but I had more than one of them. Now certainly, each chain will be unlocked. That's a known result by Cantrell and Johnson '96. '96, '98? '96. But what if I have two four chains, can they interlock? The answer turns out to be yes.

The original motivation for this proble,-- it's a neat question by itself, but the original motivation is something called Lubiw's problem. Many of you met Anna Lubiw, she's one of my Ph.D. Advisors. Last year, she was here on sabbatical. And the question is the following-- so she posed this back in 2000, a long time ago. What is the minimum number of cuts you have to make in a 3D chain in order to unlock it?

I say unlocked meaning we know they are locked chains. Suppose I want to make the configuration space connected, but I allow cheating and I allow cutting vertices. How many do I have to cut? And say the chain has n bars. Then, as a function of n, how many cuts do I need? And the answer at this point, in the worst case, we know is between about n over 2 and n over 4.

So it's at least 4 over n minus 1 over 4. And it's at most 4 n minus 3 over 2. And I'll prove to you both of those results. So we know roughly its theta n. It's in a constant factor of n, but we don't know exactly what the constant is, still a neat problem.

Originally this motivation here is, well, maybe-- there's some theories that in real proteins, maybe they disconnect their bonds for a little while so they can pass through and make things cross. How many disconnections would you need because that might require a lot of energy? The answer is a lot if you have a long protein. in the worst case of course.

Real proteins have unit edge lengths almost equal angles, so maybe the answers change in that case, but given that we don't even know whether it's a locked chain, in those situations, it's pretty hard for us to answer something more complicated like this. So let me start with the lower bound. You need to make at least n over 4 cuts almost.

Any ideas how to do that before I show you the answer? Given the one tool we have, which is the knitting needles.

AUDIENCE: Yeah, I mean if you [INAUDIBLE].

PROFESSOR: Just connect a whole bunch of knitting needles together. Yeah. Now, knitting needles has five edges, so that would naturally give n over 5. But if you're a little bit clever, you can share edges. If you share the long edges from one knitting needle to the next, and you get about 4 bars per knitting needle, except the very first bar which have to count extra, and so that's the n minus 1 divided by 4.

Now if you don't cut one of the vertices-- and here, we're considering vertex cuts-- if you don't cut one of those four vertices of that knitting needle, the whole thing will be locked because that's a knitting needle. So you have to cut one of those four, and one of those four, and one of those four just for the pieces to not be locked.

The trouble is-- so the natural algorithm, if I give you some big complicated chain, is cut every fourth vertex. That we know does not work. I think I have to wait to see that one. Let me tell you what's known about interlocking, and then we'll get back to Lubiw's problem. But at least we've proved this part. And next thing I want to do is prove this part. And to show you how we might do that, I think I need a whole board.

We're going to think about an open chain of length 2-- let me make a little more room-- 2, 3, 4 interlocking with-- start with an open chain of length 2, 3, or 4. There's going to be other results, but it's a good start.

So for 2, 2 chains, I claim are always separable. And I'm going to put a little asterisk to mean another thing is true. A 2 chain and a 3 chain is always separable. A 2 chain and a 4 chain is always separable. This matrix is symmetric. A 3 chain versus a 3 chain also is separable. But everything else, you can interlock.

I guess I haven't defined interlock. Be a good idea. I call a bunch of-- a collection of chains separable if they can all fly to infinity away from each other. So the distances between them can get arbitrarily large, otherwise they're interlocked. And usually, that will mean that no pair of them can fly away from each other. Here, we're just thinking about two at once, so it's either separable or interlocked.

So this is a worry if you're thinking about cutting this apart. You cut every fourth bar-- fourth vertex, if I get the counting right, I think that means you have length 4 chains connecting them. Now, we know length 4 chains can't be locked, but they could be interlocked. And so you could actually build an example where if you do the simple pattern of every fourth cut, you will get interlocking. It's still open whether you could be clever and still only make about n over 4 cuts and get everything to separate, but to solve that, you'd have to get around this boundary, this annoying situation, which we don't fully have characterized. We just have examples.

And we also thought about-- and we'll get to this later-- if you try interlock an open chain with a closed chain. With two closed chains, it's trivial. You can just make a knot, or make a link. I guess is the proper term. But an open versus a closed is more interesting, especially if that closed guy is not itself a knot.

I think up to length 5. Yeah, I think the smallest, locked, closed chain is of length 6. Never showed it to you, but that is a known fact. So up to 5, we know this thing is not by itself locked, so it's not really cheating compared to an open chain. Then, the answer is separable, separable, separable.

It's really hard to lock things with a 2 chain. 2 chains, we call hairpins. That's a 2 chain. Everything else is interlocked. Basically, the same results hold whether one of your chains is opened or closed because notice there's a shift here. The smallest closed chains have length 3, and the smallest open chain is really of length 2. I guess you could consider single bars, but they don't usually do much. You're can just slide them out parallel to themselves.

So there's a shift in the indices here, and that's why there's a shift in the matrix. Sounds ominous. A shift in the matrix, what will we do? But they're basically the same. So I'm going to show most of these results to you. The stars mean that even if I give you-- get this right-- even if I give you a whole bunch more hairpins-- so for this result, it's the most interesting. I take a 3 chain, a 3 chain, and then a millionaire hairpins, a million 2 chains. Still, they can separate. And all the results with stars, you can add arbitrarily many hairpins.

These two are the most interesting, a 4 chain with any number of hairpins, a 2, 3 chain with any number of hairpins. These results do not hold or we don't know how to prove it anyway. That's the summary.

There's one question that's not answered by this table because it's how low can you go and get interlocking. This is saying, well, once you get to a 3 chain and a 4 chain, you can interlock. It's the minimum. But what if you have a bunch of 2 chains or even just one? Can I interlock it with some chain of some length?

Let's say, I mean up to 4 is the most interesting because once you get to 5, you can have knitting needles. But suppose I take longer open chains down here, but I require that they're not themselves locked. Can I interlock it with a hairpin? It has practical applications to hair I guess. And the answer is yes. So this first example is a 16 chain. That's the gray part. Wow! It's hard to even follow the ends. Here's one end. And here's another end and 16 bars in the middle. And then the red thing is a 2 chain, and this is a model of that thing, physical model. And it's locked.

Sorry, it's interlocked, I should say. I'm pretty sure you take either of the chains, obviously with the hairpin, but also for the really complicated 16 chain, it will unfold by itself. But with a hairpin, it is held open at that angle and can't unfold. The best bound so far is an 11 chain versus a 2 chain.

It's conjectured to be optimal. There's various reasons to believe at least this interlocking trick where you-- basically, all the hairpin can do is hold an angle open or try to. And this seems to be the smallest way to interlock it up there and the smallest way to interlock it on either side. And these are zooms. This is of one of these corners. This is of the top part.

So those, by themselves, if you're trying to do these three interlock tricks, these are minimal, we think. But as an assembly, it's conjectured to be optimal, but we don't know how to prove that. But at least these guys are interlocked, so 11 chain with a 2 chain.

So those don't fit on the table because the table's not large enough. But let me start with this result. So the easiest thing to do is show that if I have any finite set of hairpins, 2 chains, they never interlock. This is actually an old result although it was never phrased this way until earlier this decade, I guess, by de Brujin, a famous Dutch mathematician from 1954.

He proved essentially this. He proved a simpler version which was then generalized in 1984 by Robert Dawson in Canada. And they proved something more general. They said, well-- I will say a hairpin is star-shaped, meaning if you were right here, here's your eye at this point, you can see the entire interior of the shape without leaving the shape. So that's a star-shaped figure.

And these guys proved that for star-shaped figures, there's a way to always separate them without even changing any of the angles. There's no folding involved. You could think of rigid 2 chains. They can all fly apart from each other. How do you do it? Some fun geometry. You say, I have my favorite 0, 0, 0 in three space. I just want to scale everything from the point, make space bigger.

What will happen to these poor little chains? They get bigger. They'll get longer. As they get longer, just cut them off again. But the result is that these points will fly away from each other. And so all of these two chains are going to separate. The lengths get longer, but I can always cut them shorter. And if I scale space, obviously if I didn't have self-intersection before, I still won't have self-intersection.

So it's a fun motion. We tried to draw it in the book. It's a little hard to draw. So we have the red, green, and, blue-- hopefully, you're not colorblind-- hairpins hanging out. They're not hitting each other. Here's the origin. We scale everything, so the blue guy goes to the dashed blue. Green goes to the dashed green. It looks weird. It's all over the place. And you're stretching for this point, so as you go farther away, it's bigger. Red guy expands to the dashed red guy.

What's not shown here is then you clip the red things. Because everything gets longer, you can always clip to the original lengths. And that's a motion of these chains. You're not changing the angle. Scaling preserves angles. And everything flies way for each other. So a very nice trick from a long time ago.

Once you get to 3 chains, this doesn't work. And so if we want to prove this result, which would subsume these guys, we need to do a little more work. Although a very similar trick works. So let me show you why two open 3 chains separate. Even if you add arbitrarily many extra hairpins-- that's the star-- Two 3 chains always separate.

I guess suppose we're in a generic situation. Nothing aligns that shouldn't. I have one 3 chain, or something like this. And I have another one. Let's make it interesting. Is that possible? Maybe. Something like that. Probably you want these guys to be pretty long. Do some crazy whatever. That is two 3 chains if I drew it right. And the claim is you can always separate them.

So here's the deal. I have this middle bar. There's a whole family of planes that are parallel to that bar, in fact, a lot of planes. There's what? Three dimensions of planes. There's a two dimensional family of planes that are parallel to that bar. There's a two dimensional family of planes that are parallel to that bar, meaning if you extend it, it doesn't hit the plane or lies in the plane.

So there's actually a plane that is parallel to both bars. In fact, there's a whole one dimensional family of that, if I figured it out right. So you can actually find a plane that bisects the two bars. So there's one on one side, one on the other side. And even if the bars extended, it would not hit the plane. So there's some plane in the middle here. Really hard to draw in 3D.

Actually, I think I have a picture. It might be a little easier to see. So it looks something like this. You have to believe me. This is general case. You have the red chain and the blue chain. The middle bars are on opposite sides of the plane, and they're parallel to the plane. Now what we do is scale, but we only scale in z, so non-uniform scaling. This will not preserve angles. Here, we're going to fold.

So the result, we scale from z equals zero, so these guys get pushed down. Those guys get pushed up. Scaling in z, again, does not cause any self-intersection. And because we made these guys parallel, these edge lengths will be preserved. That's the key. The end bars will get longer, but I don't care if they get longer. I can always cut them shorter. So that's the trick. Yeah, they'll always get longer, even if they don't go to the other side.

So that's how you can just pull them apart. And also, if there are 2 chains in here, they will also just fly away from each other. So eventually, everything will get far away unless they happen to have the same z-coordinate, but then it's degenerate. Just perturbed a little bit. Jason?

AUDIENCE: Is this the case where the two bars are coplanar and parallel?

PROFESSOR: Yeah. I said at the beginning generic, so in particular, I want to assume that these guys are not coplanar, otherwise, you could just perturb it a little bit. As long as they're not touching, you can wiggle everybody. Good point. That's two 3 chains. There's one more result in the separability. We've done all of these separability results here.

The next one would be this, which is the same as this. So here we have a 4 chain versus a 2 chain. Basically, the same trick works. I just have reorient to put middle bars in the xy plane. So in other words, there's two middle bars. They live in a plane because they share a vertex. So think of that as the plane and then scale perpendicular to the that. Same trick works.

So 4 chains versus various 2 chains. Again, we preserve the lengths of the two middle bars because they lie on the plane. So there's no scaling that happens in the plane, and in the generic case, everybody will be off the plane, and they fly. Cool. So that proves all of these separability results. I'm not going to do these closed chain ones here. You can read the paper if you're interested.

At this point, we can go back to Lubiw's problem and prove this result. So basic algorithm is cut every other vertex. You'll get a whole bunch of hairpins. Hairpins, we know, separate. But we can be slightly more efficient by leaving an initial segment of 4 chain or two 3 chains. I think it comes out to the same number. And you end up saving three if we computed this right. So you get n minus 3 over 2.

I think our first bound on this problem was n minus 1 over 2, and then we improved by 1 although I have to n minus 3 over 2. That has stood the best bound since. Basically, because with interlocked stuff-- after that, you get interlocked. And if you're going to avoid interlocking, you have to be clever. You can't just use the numbers.

So that is all we know about Lubiw's problem. Now, I get to show you all the fun interlocked examples. We're going to do pretty much all of these. So we start with one of the prettiest examples, the threefold symmetric. This is three 3 chains. So that's not even drawn here. Because this table is about pairwise interactions.

We know that two 3 chains-- we just proved two 3 chains cannot interlock, but three of them can. That's the only case where that question is interesting. So everybody else, you can interlock. Unless you have 2 chains, and we know those don't help. So three 3 chains interlock. I'm not going to try to prove that here. It's a pretty complicated argument where you look at the center edges. You look at the convex whole of those edges and argue these guys can't stick into the center. It's a geometric argument. If you're interested, check out this paper.

It's a pretty cool example. This was a great-- finally, I had a good exercise for using Rhino yesterday and drawing this figure. It was a lot of fun. This is a 3 chain and 4 chain. One, two, three, four. Just two different views of the same thing. And that proves this result, which is this result.

It's pretty close, pretty tight. Again, it's a geometric argument. It's tricky. I'll go into it here. We found these by building lots of straw models and jiggling them around with string connecting the straws together. But then we had to prove it. For a while, we weren't sure whether a 3 and a 4 could interlock. I think first we had a 4 and a 4 or maybe 4 and a 5, and we kept working our way down. And now we know this is the smallest you can get.

And now we go into the closed results. Because 4 and 4, that's of course easier than 3 and 4. So that's done. This is contained in those results. Now, we look at the closed examples. So this is going to be a closed triangle interlocking with an open 4 chain. And here I'm going to give you some of the details because this proof-- it's relatively easy to prove that this is interlock. You can use a more topological argument, which is what we're used to from the knitting needles.

Remember we took a ball that separated the interior vertices from the endpoints. We said the endpoints really can't get in if we set the ball right. And therefore, you could connect the endpoints by a big rope, and then you have an Trefoil knot, and there's no way you can untie the knot with that having intersection somewhere. You can argue the rope never self intersects, so it must be the chain.

We can do the same thing with this example, with a 4 chain interlocked with a triangle. And we'll sketch that proof. So I want to prove that those two guys are interlocked, so suppose that somehow they separate from each other. If there's a separating motion, there's a motion where the triangle actually doesn't move at all. So triangle's rigid, except that it can move by rigid motion, translation and rotation.

But any motion I have, I can recenter things by relativity so that the triangles stays fixed and just the quadrilateral moves. So that's nice because a triangle is a natural thing to draw a sphere around, and I want to argue if these bars are really, really long, the endpoints will be outside this big sphere. So let me tell you how long.

I have my triangle I'm going to draw a sphere that contains the triangle. Say this sphere has radius little r. Big R is going to be little r plus the sum of the middle bars, which should sound familiar from what we did with knitting needles. This is of the 4 chain. There's two middle bars. Add up their lengths. Add on r. This is just a big number, capital R.

I'm going to make it bigger. So I have my little triangle. It has a tiny ball around it. I'm going to draw a pretty big sphere here of radius 15 r, an even bigger sphere of radius 20 r. And I don't need to be precise about exactly where the center is, but I guess it would be the center of this ball.

Here's what I want to say. I'm supposing the lengths of these edges is more than 20 r. So initially, those bars go outside the box. The endpoints are outside of the big ball. Now, here's the claim. As long as the endpoints of the really long-- the endpoints of the 4 chain stay outside this ball, then the interior bars-- these two middle bars-- have to stay inside this ball. This is just like the knitting needles.

As long as the super long bars, the endpoints, stay outside this ball, the middle bars have to stay inside. That's just because those bars are so darn long. And so for as long as that holds, you can compute the topology-- I'm not going to try to draw it-- the topology of-- in this picture, if you connect to the ends of the purple guys, because they're outside the big ball, the 15 r ball-- not sure I really need the 20 r ball, but the 15 r ball is the key-- topology you get is this.

That is the link between the circle, which is this nice loop. And the purple guy is that red loop if you just figure out what it looks like currently. And as long as the endpoints of these guys don't go inside and those guys remain on the inside, basically until something interesting happens, the topology has to be that. So at some point-- sorry, I said that slightly wrong.

As long as the middle bars stay inside the ball and the endpoint stay outside the ball, a 15 r, this will be the case. They are not saying nothing interesting happened. So violation, the only way to open this thing could be an endpoint goes inside or a middle bar goes outside the ball. Consider the moment when that happens. If a middle bar, one in the middle vertices is going to go outside this 15 r radius ball, then you know that all of the middle bars are far away from this triangle because those edges very small compared to capital R.

So we're 15 away. If you have one middle vertex that's outside, all of them are pretty close to outside. They're all going to be in some smaller ball here-- outside that smaller ball. The alternative is that one of the end-- that's not a contradiction yet, but we'll come back to it-- the alternative is that one of the endpoints of the bars goes inside this radius 15 r ball.

When that happens, because this bar is so long, the other end of the bar will actually be outside the sphere. And so then, again, all the middle guys are outside the sphere. So either way, you violate. I assumed this before, but now, I'm saying if you try to take these guys out or if you try to push this guy in, the middle guys are outside the ball already. So consider a situation when the middle edges are all outside the ball of radius 15 r.

The only thing that could penetrate some tiny sphere, like this one of little r, only parts of the 4 chain that could penetrate that are the long bars because the middle stuff is too far away. The long bars are super long, so they could conceivably penetrate the triangle. All we really care about is piercing the center of the triangle because that's when we get interesting topology.

We want to get this topology, but sadly, you could only get these topologies. I guess luckily. You can only get those three topologies, which don't match this one. And so there's no way to get continuously from one of these to the other. That's the argument. Let me say a little more which is if neither of the two long bars hit the triangle, then you trivially get this. You don't get any linking.

If one of them strikes a triangle, you always get this. If two of them strike the triangle, it's possible to get this or this depending on how you link it up. And I'm not going to prove it here, but that's all you can get. And so if you can't get this one, you're in trouble. There's no continuous motion from this to those without crossing.

So that's how it goes. And this is an example of a topological argument. You can use the same kind of argument to prove this last result, which is this one. But now this is not necessarily symmetric because one's closed, one's open. So this is a closed 4 chain quadrilateral, in other words, with an open 3 chain, which can also lock.

And again, you can use the topological arguments. A little messy because now this is like a tetrahedron instead of a triangle, but the same tricks work and topology, luckily, doesn't match between the inside and outside cases. Cool. So that is interlocking and pretty much the entirety of this table.

Now, there's a way to generalize this table, which is instead of saying, oh, I have universal joints. Everything's nice. I start thinking back to proteins and say, well, what if I had some fixed angles. What if some of the chains are fixed angles and others are universal? What if some of the chains were ridge? Because we know that rigid hairpins can separate without any folding whatsoever, what if some of them are rigid and some of them are fixed angle and some of them are universal?

Well, we haven't done all of those results, but most of them for pairs, we have got universal and rigid. I mean a fixed angle 2 chain is a rigid 2 chain, so there's only two columns there. Universal fixed angle rigid. Universal fixed angle rigid. And here we only get up to rigid. Yeah, I'm not sure if it makes a huge difference out there.

Again, this is not a complete story. There other results that might fit in here. How far down do those minuses go if you don't have rigid parts? But let's see. The plus means there's locked things. The minus means they always separate. It's a little more concise. And we've sorted the rows and columns so that you get nice-- conveniently you get a nice diagonal pattern, not perfect diagonal, but it's at least monotone.

It makes sense that a rigid-- obviously, rigid is more fixed than a fixed angle. But it's not obvious that a universal 4 chain is going to lock more than a fixed angle 3 chain, but turns out that is the case for these examples. Now, you could consider triples of them, each with different properties. Then, you get a three dimensional table. That gets a little messier.

Probably not a huge number of results are necessary along those lines. Here, we just needed the 2 chains plus the three 3 chains were interlocked. So there's more work to be done here, but this was a lot of fun. And if you want to see the examples, you should check out this paper. This was basically for fun and to see how low could you go in terms of number of edges and various combinations of different parts. But it turns out this interlock stuff does have applications. I'm not going to call them practical because it's about another theory result.

And its result, I mentioned. If you have some 3 chain-- oh sorry. Some three dimensional chain-- let's say universal joints-- and you have two configurations, configuration a and configuration b, and you want to know is there a motion from configuration a to configuration b. This is PSPACE-complete, result by a bunch of Berliners from a few years ago.

And basically, what they do is build a computer. I don't have the gadgets here to show you. But they build a computer with various moving parts and what not. To build the infrastructure of that computer, the rigid part doesn't move. They use interlocked chains. Basically, if you want to build a nice, rigid framework, like a truss, you can do it because you just take chains.

I'm only allowed to make chains. I'm not allowed to make graphs. But if I bring two parts of the chain really close to each other, I can basically tie a knot by making lots of little links and using say an interlocked 3, 4 chain, interlock 3 chain with an interlocked 4 chain. In the same way when we're building the locked 11 chain with the lock 2 chain, you're tying little knots around various points.

Here, we don't even have to be efficient, just constant numbers are fine. So you can build some rigid structure like a tetrahedron by taking one path here, making an interlock structure, making interlock structure, interlock, interlock. Could double traverse, I guess. And you can interlock everything using a chain. Build a rigid structure. Then, you go and build the flexible part. And that's the fun computation.

But just to get off the ground and to build some rigid infrastructure, to build gadgets basically in your hardness proof, they use interlocked chains, which is pretty cool. This is a really interesting result, a very negative one. It's really hard to fold from one shape to another, but it uses this. They also prove the same thing for trees in two dimensions.

For that, they used the lock trees-- which we covered way back, lecture three or whatever-- these lock trees to build-- to basically fill rigid angles, to force things to be at particular angles, and then they have flexible parts to do the computer stuff.

So it's fun. They use our little examples to build big computers. Still open, of course, is can I start from a straight configuration of the chain and can I get it to this.

So in other words, given a chain, can I unfold it. We'd like to know whether that's polynomially solvable or empty complete. It is about characterizing lock chains in some sense. This is about going from arbitrary configuration to arbitrary configuration. And here, you're allowed to build rigid infrastructure which never moves. Here, everything would have to fall apart somehow, so you can't use interlock chains. You can't use lock trees in 2D.

So this problem is still open. Seems quite difficult because you can't build any infrastructure the stays around forever. Everything would have to fall apart if your machine terminates. Any questions? Wanted to talk about that for a long time. Now that we have interlock chains, you get to see a little bit behind the scenes of what's happening. If you want to see the full proof, check out that paper by [INAUDIBLE], [INAUDIBLE], and [INAUDIBLE]. All right. That's it.