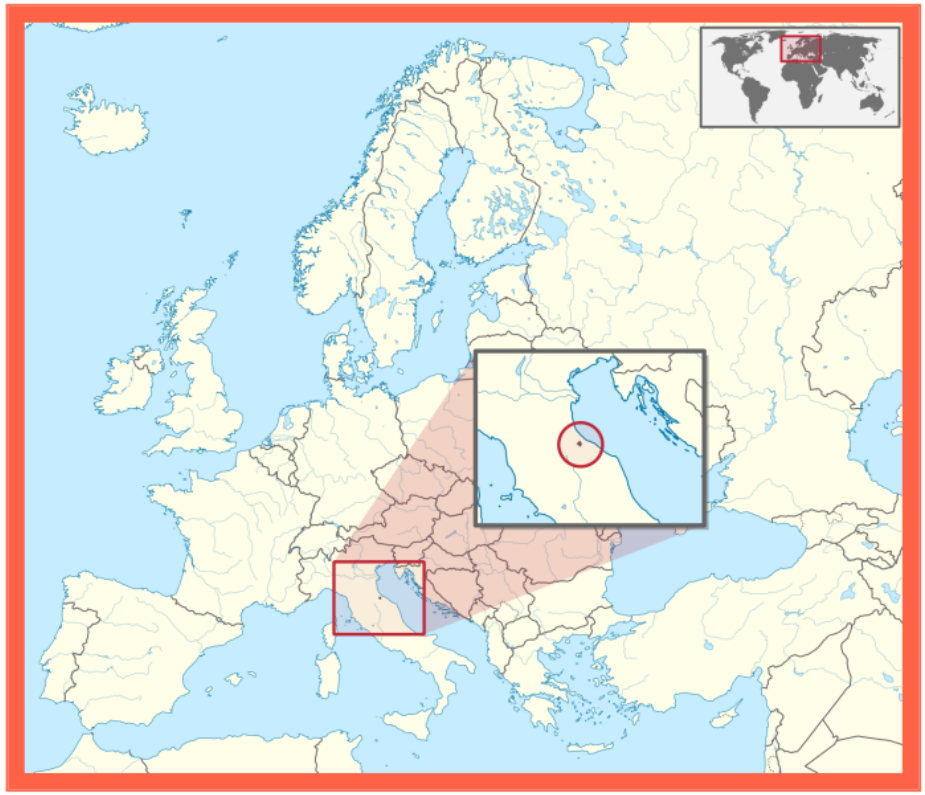
An unknotter

Or: Local versus global

~~Accept~~Change what you cannot ~~change~~accept

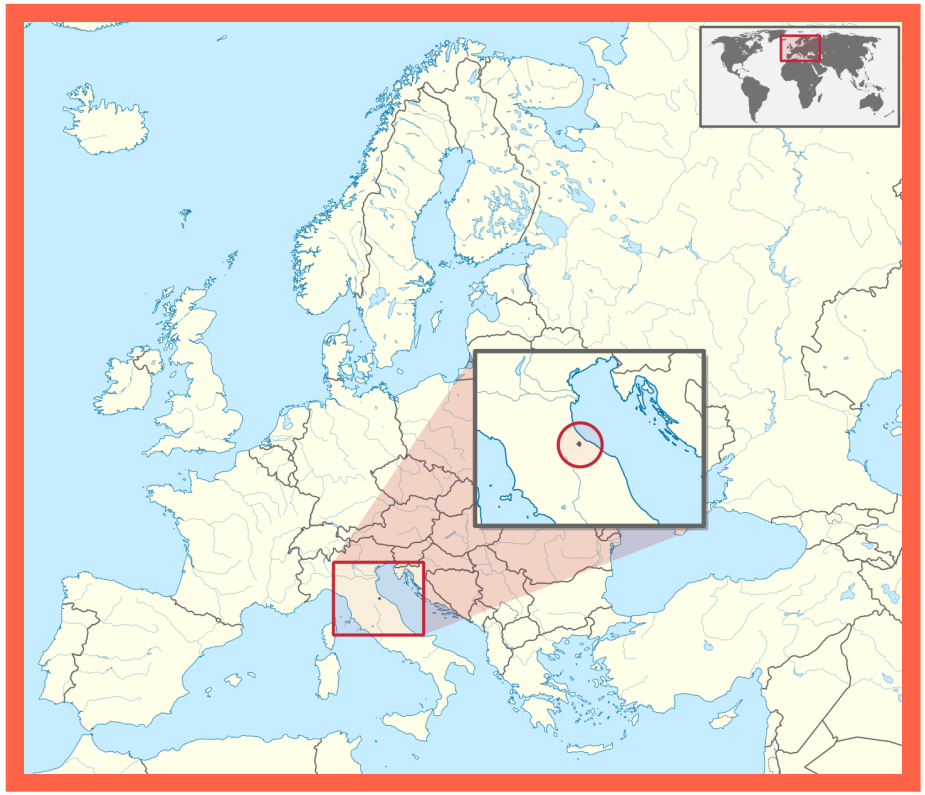


I report on work of many people (Brittenham, Hermiller, Applebaum et al. (sorry too many), Dranowski, Kabkov, ...)

An unknotter

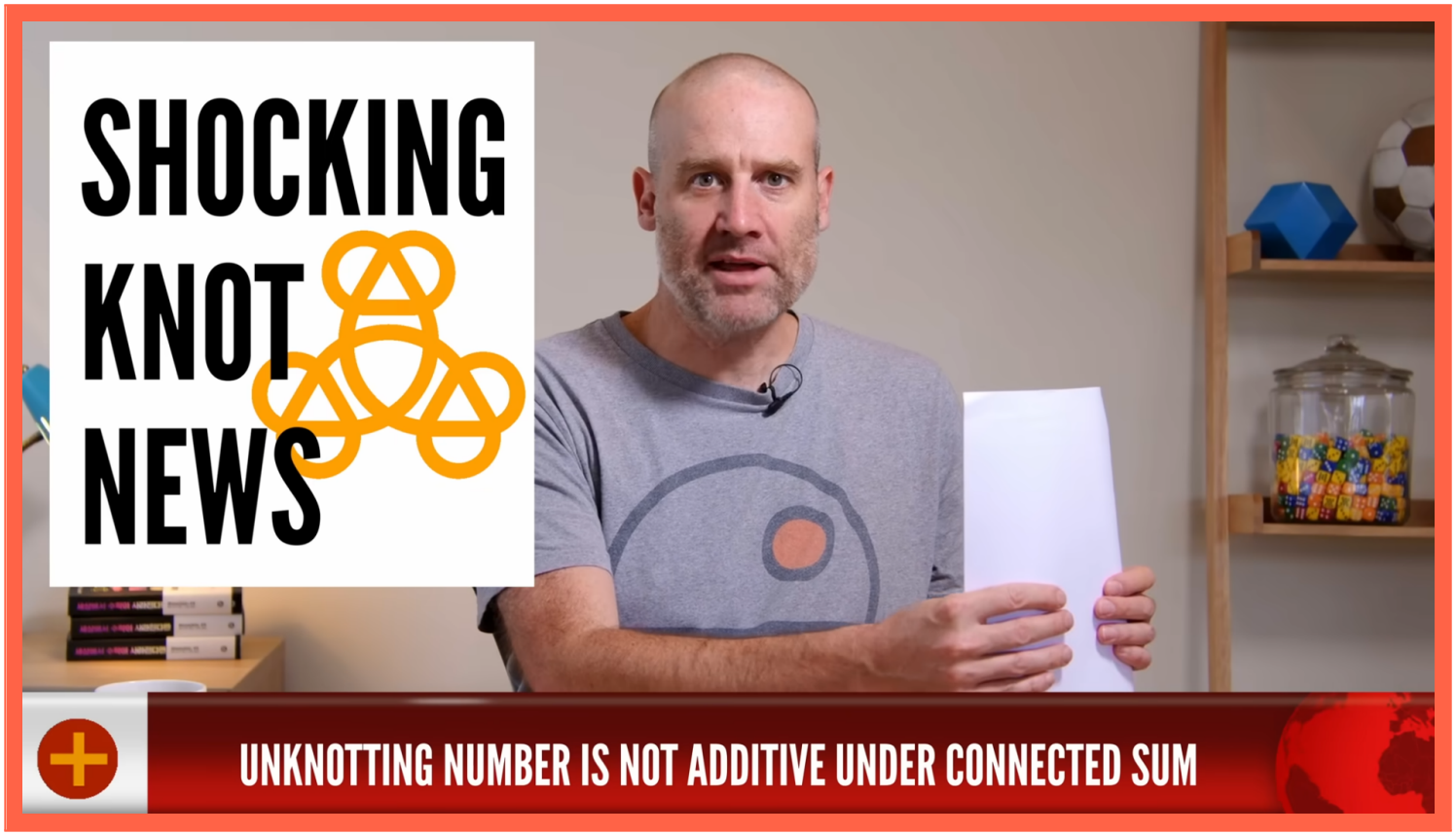
Or: Local versus global

~~Accept~~Change what you cannot ~~change~~accept



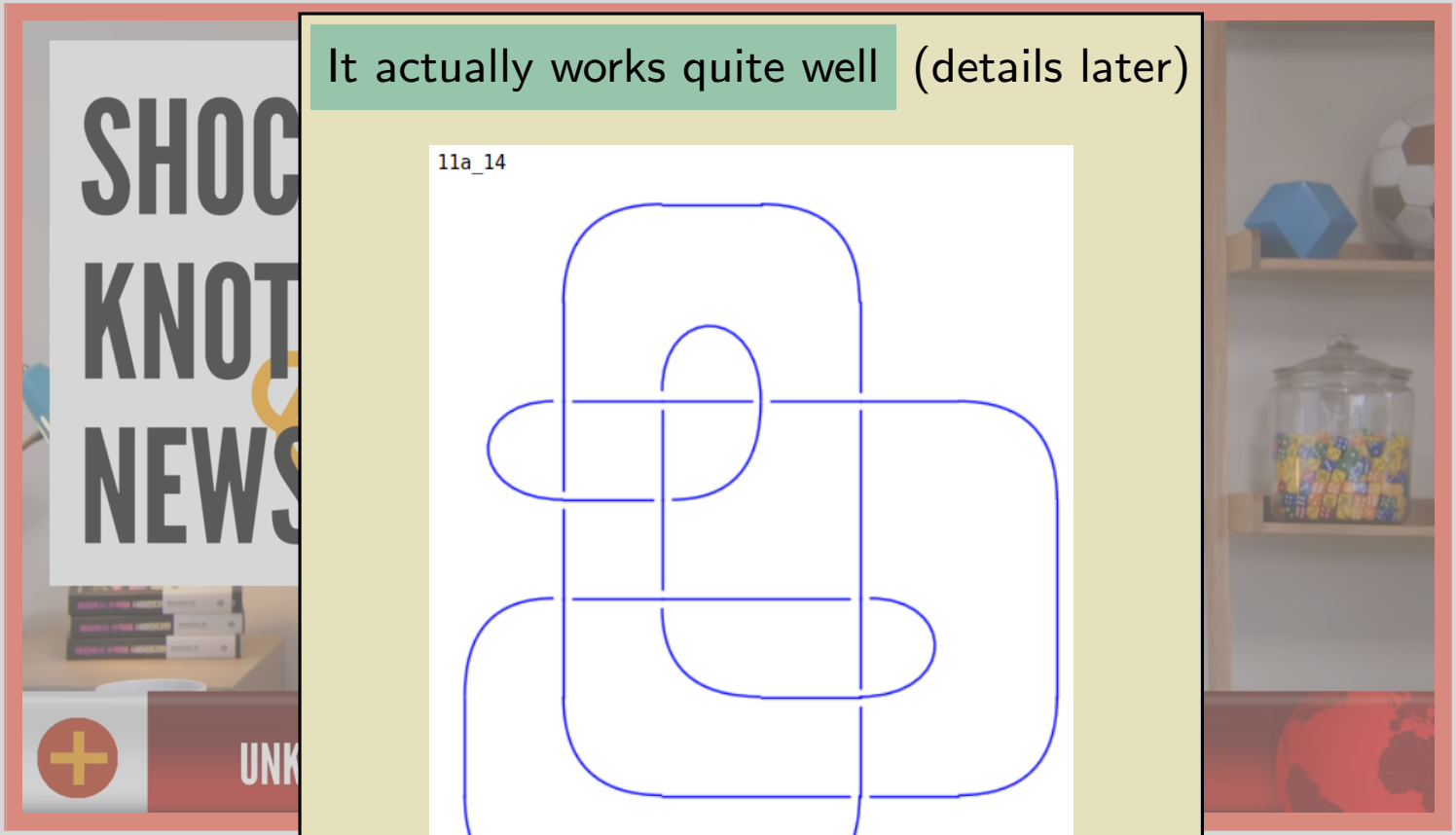
I report on work of many people (Brittenham, Hermiller, Applebaum et al. (sorry too many), Dranowski, Kabkov, ...)

Unknotting is hard



- Amazing Standupmaths https://www.youtube.com/watch?v=Dx7f-nGohVc
- My take away You really do not want to do this by hand :-)
- Idea Let a machine do it

Unknotting is hard

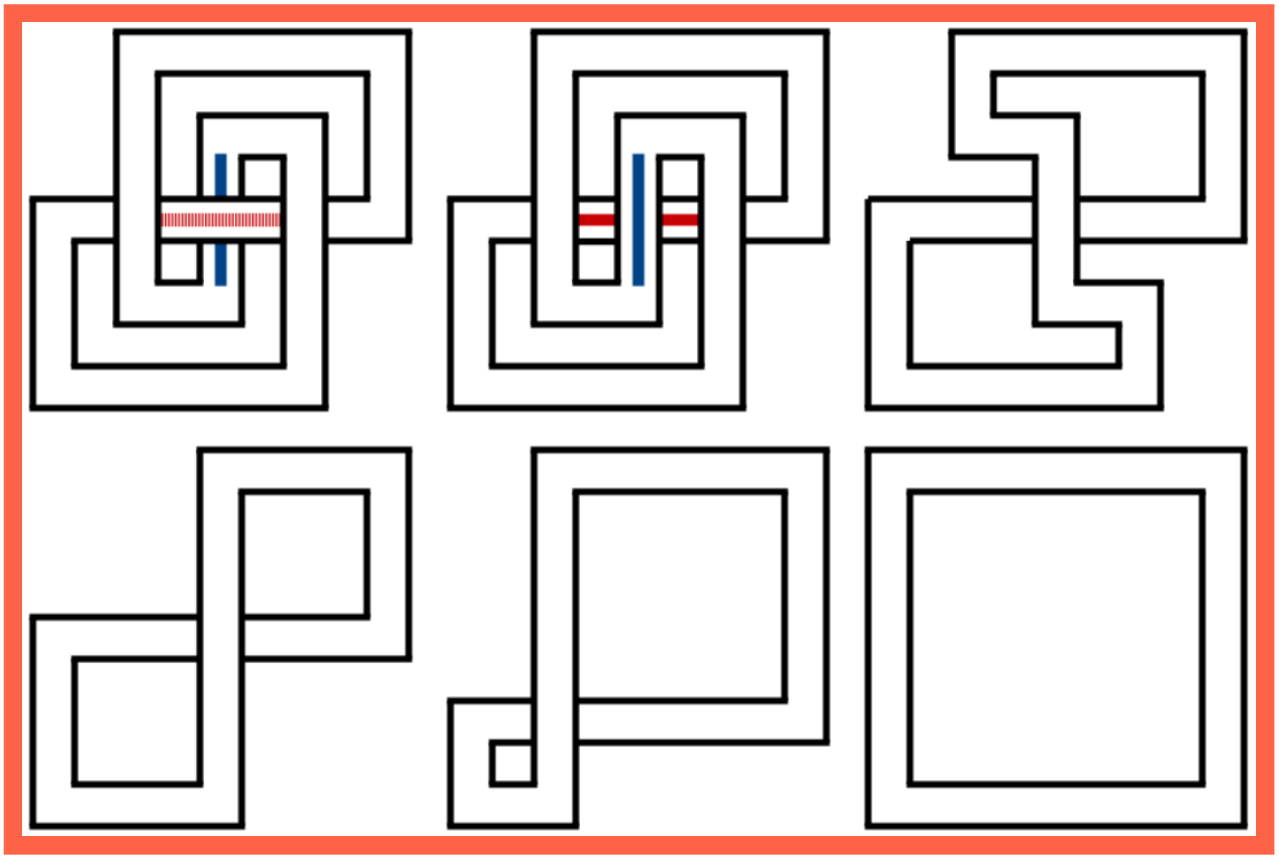


It actually works quite well (details later)

The smallest knot for which it found
a nontrivial reduction sequence
is the 11 crossing knot above

- Amazing Stanc tch?v=Dx7f-nGohVc
- My take away d :-)
- Idea Let a machine do it

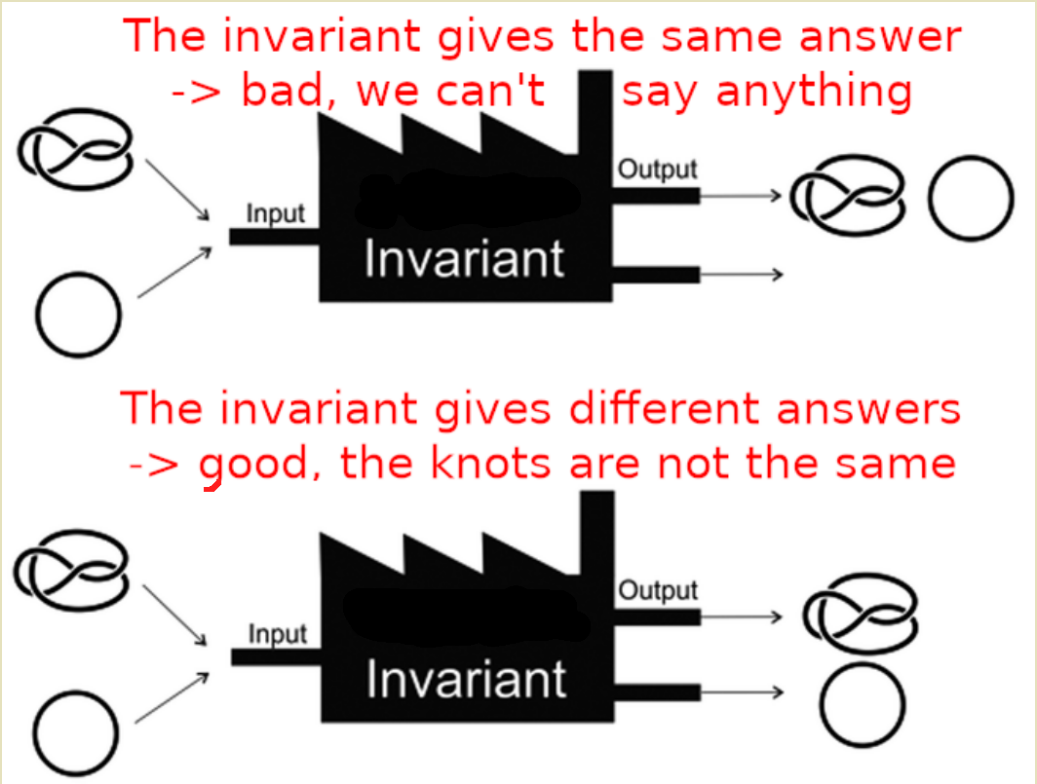
Unknotting is hard



- Unknotting number u = minimal number of crossings flops needed to reach the unknot
- Dream You can find u on a minimal crossing diagram
- Problem (why this is so difficult, part a) That is not true

Unknotting is hard

Problem (why this is so difficult, part b)

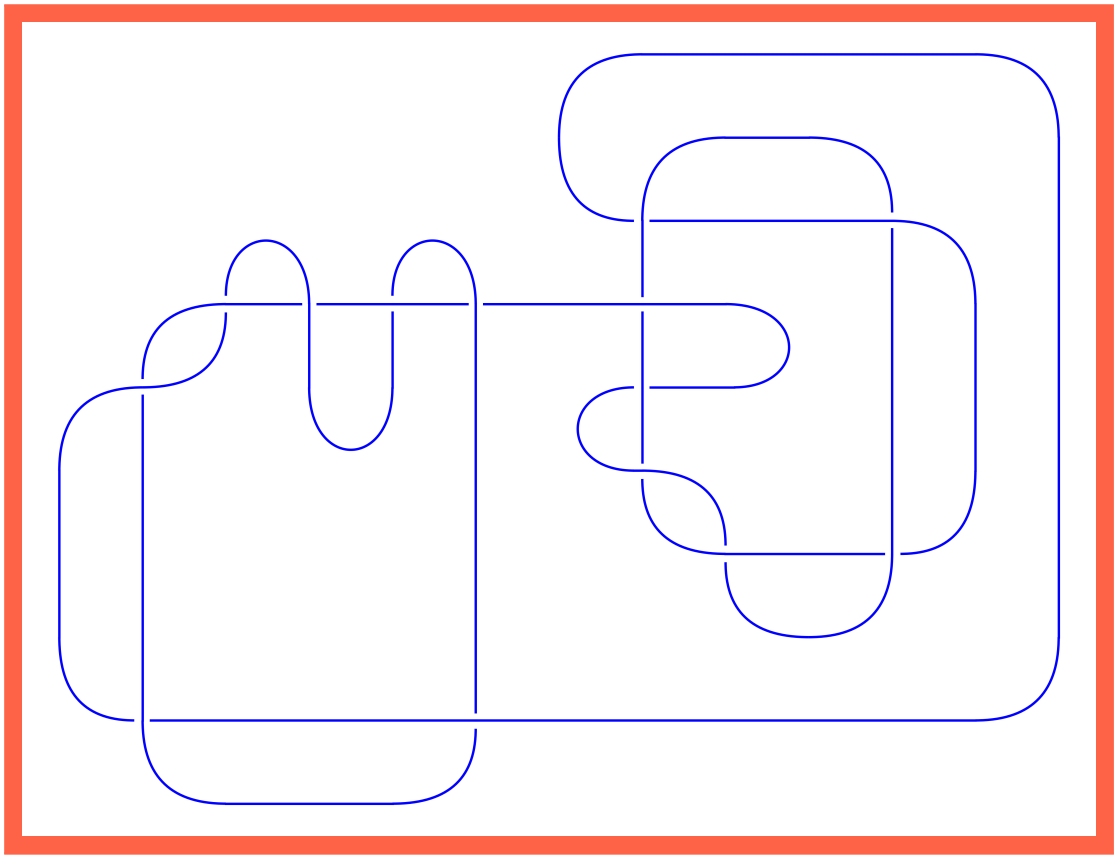


I didn't know any "good" invariant
that gives "interesting" information about u

Side quest With details to come, we can change that!

- Unknotting … ach the unknot
- Dream Y…
- **Problem (why this is so difficult, part a)** That is not true

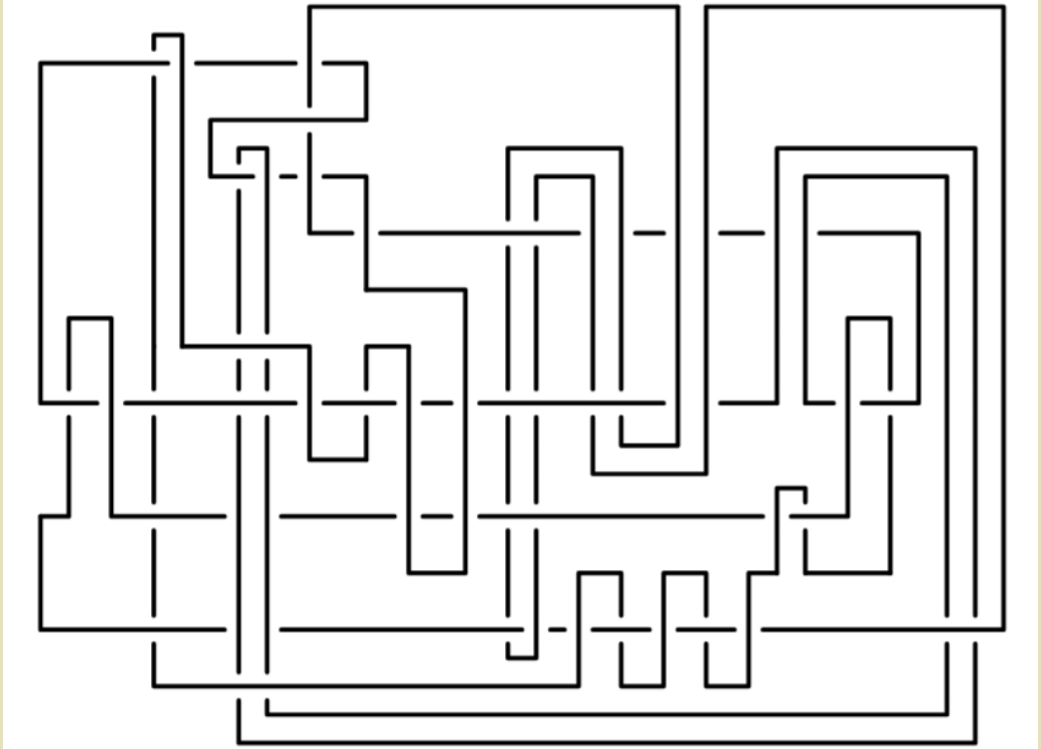
Unknotting is hard



- Open for a long time Unknotting number is additive under connected sum $\#$
- No! **Brittenham–Hermiller 2025** ("obvious" diagrams hide the real unknotting)
- Key What I am going to explain is a variation of their ideas

Unknotting is hard

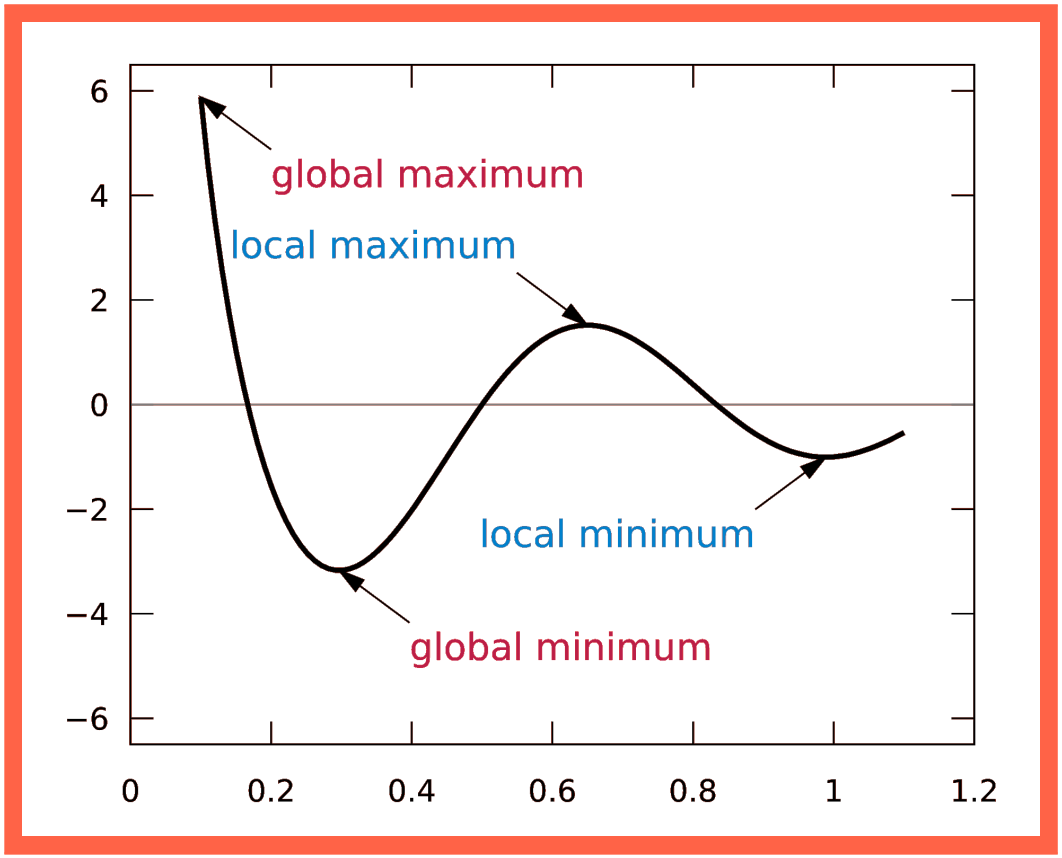
For $7_1 \# \mathrm{mir}(7_1)$



The minimal known diagram with $u = 5$
has 56 crossings (**Wang–Zhang 2025**)

- Open for a lo… connected sum #
- No! **Brittenham–Hermiller 2025** (…"obvious"…diagrams hide the… real unknotting)
- Key What I am going to explain is a variation of their ideas

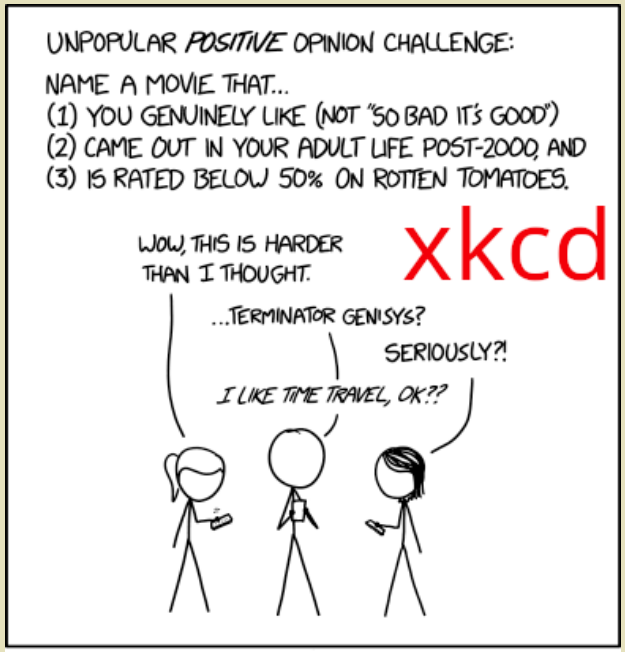
Unknotting is hard



- Problem a (rephrased) Human intuition is local; unknotting often is not
- Problem b Friendly invariants have not much to say about u
- Idea A perfect task for AI '=' reinforcement learning (RL)

Unknotting i

Why is this perfect for AI?



1) There could be a subtle pattern
(not detectable by humans) that AI could exploit

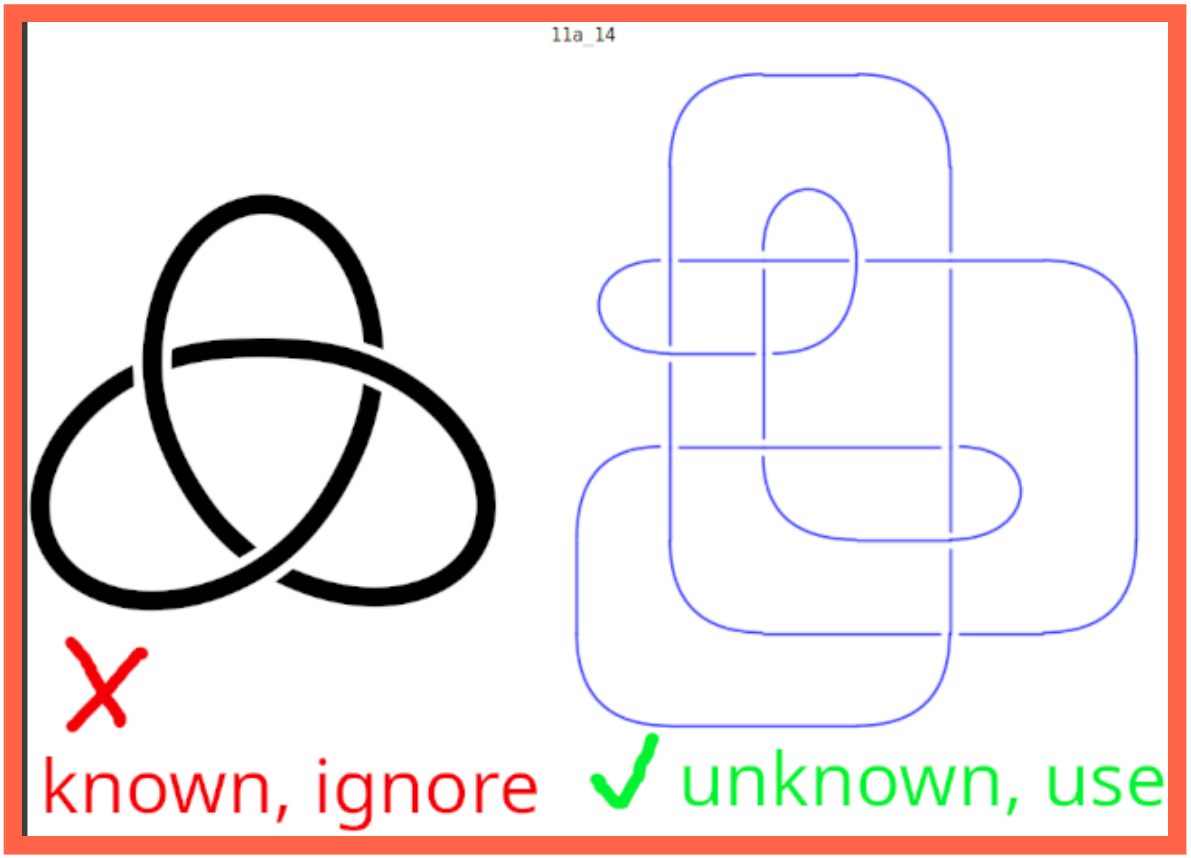
2) Repeating the search a million times is likely helpful

3) This is really not something you want to do by hand :-)

- Problem
- Problem
- Idea A perfect task for AI = reinforcement learning (RL)

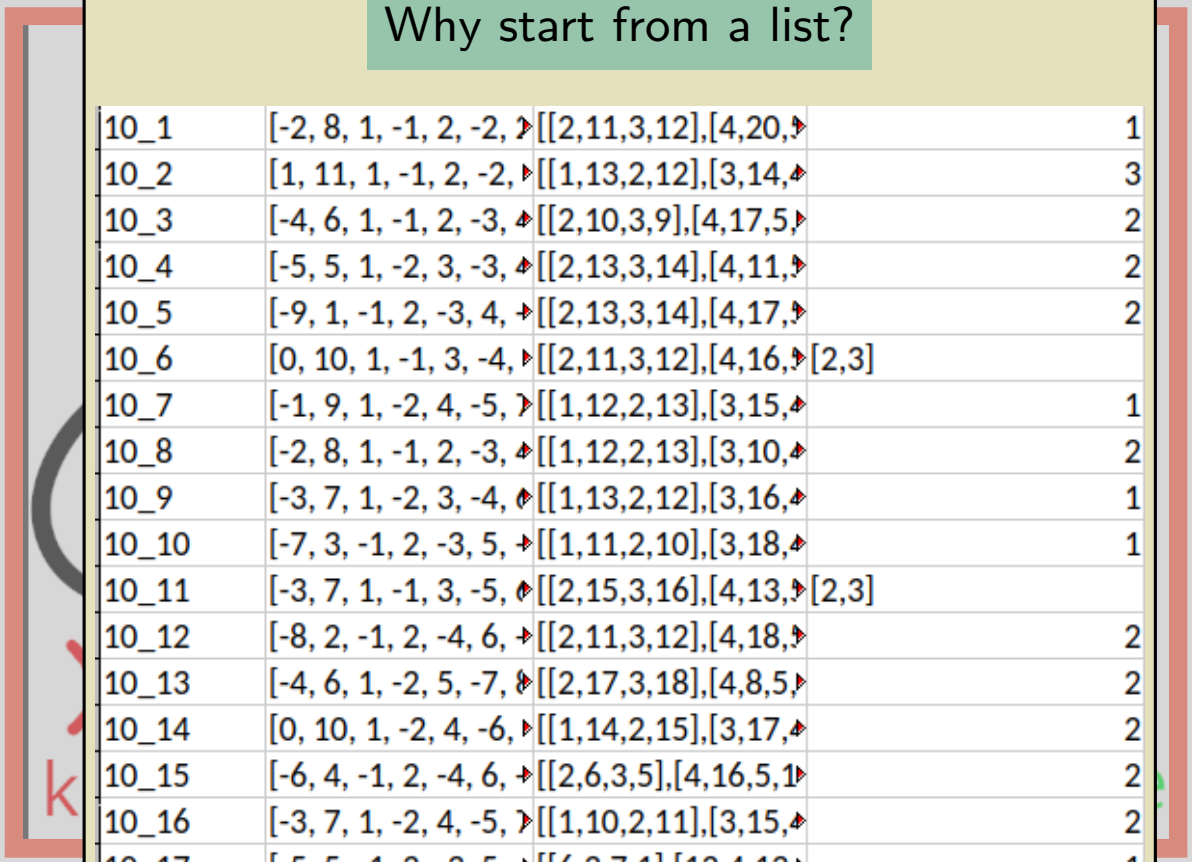
ten is not

The pipeline



- Input Take a knot diagram from a list (when u is not known)
- Goal Improve the upper bound by finding a better route to the unknot
- Key Do not insist on working with the smallest visible diagram

The pipeline



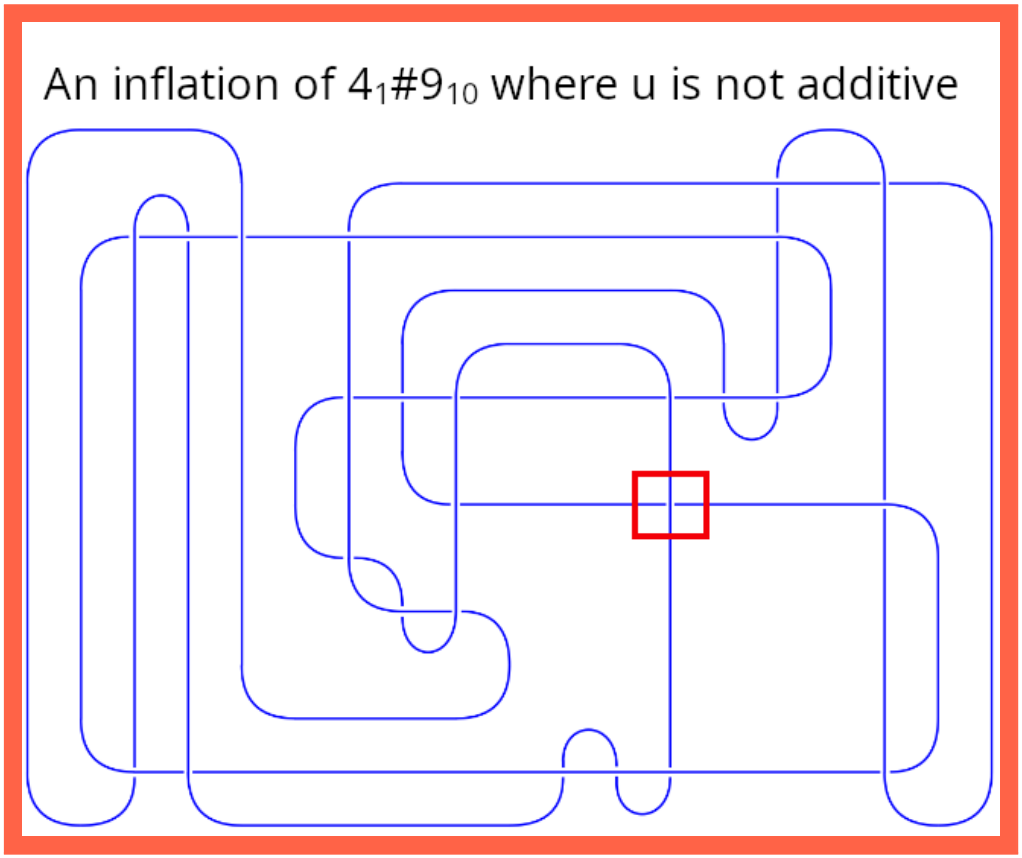
Why start from a list?

10_1	[-2, 8, 1, -1, 2, -2,	[[2,11,3,12],[4,20,	1
10_2	[1, 11, 1, -1, 2, -2,	[[1,13,2,12],[3,14,	3
10_3	[-4, 6, 1, -1, 2, -3,	[[2,10,3,9],[4,17,5,	2
10_4	[-5, 5, 1, -2, 3, -3,	[[2,13,3,14],[4,11,	2
10_5	[-9, 1, -1, 2, -3, 4,	[[2,13,3,14],[4,17,	2
10_6	[0, 10, 1, -1, 3, -4,	[[2,11,3,12],[4,16,	[2,3]
10_7	[-1, 9, 1, -2, 4, -5,	[[1,12,2,13],[3,15,	1
10_8	[-2, 8, 1, -1, 2, -3,	[[1,12,2,13],[3,10,	2
10_9	[-3, 7, 1, -2, 3, -4,	[[1,13,2,12],[3,16,	1
10_10	[-7, 3, -1, 2, -3, 5,	[[1,11,2,10],[3,18,	1
10_11	[-3, 7, 1, -1, 3, -5,	[[2,15,3,16],[4,13,	[2,3]
10_12	[-8, 2, -1, 2, -4, 6,	[[2,11,3,12],[4,18,	2
10_13	[-4, 6, 1, -2, 5, -7,	[[2,17,3,18],[4,8,5,	2
10_14	[0, 10, 1, -2, 4, -6,	[[1,14,2,15],[3,17,	2
10_15	[-6, 4, -1, 2, -4, 6,	[[2,6,3,5],[4,16,5,1	2
10_16	[-3, 7, 1, -2, 4, -5,	[[1,10,2,11],[3,15,	2
10_17	[-5, 5, -1, 2, -3, 5,	[[6,2,7,1],[12,4,13,	1

One targets knots with a known lower bound
but still no sharp upper bound in hand
so every success gives genuinely new information

- **Input** Take …vn)
- **Goal** Improv… …the unknot
- **Key** Do not insist on working with the smallest visible diagram

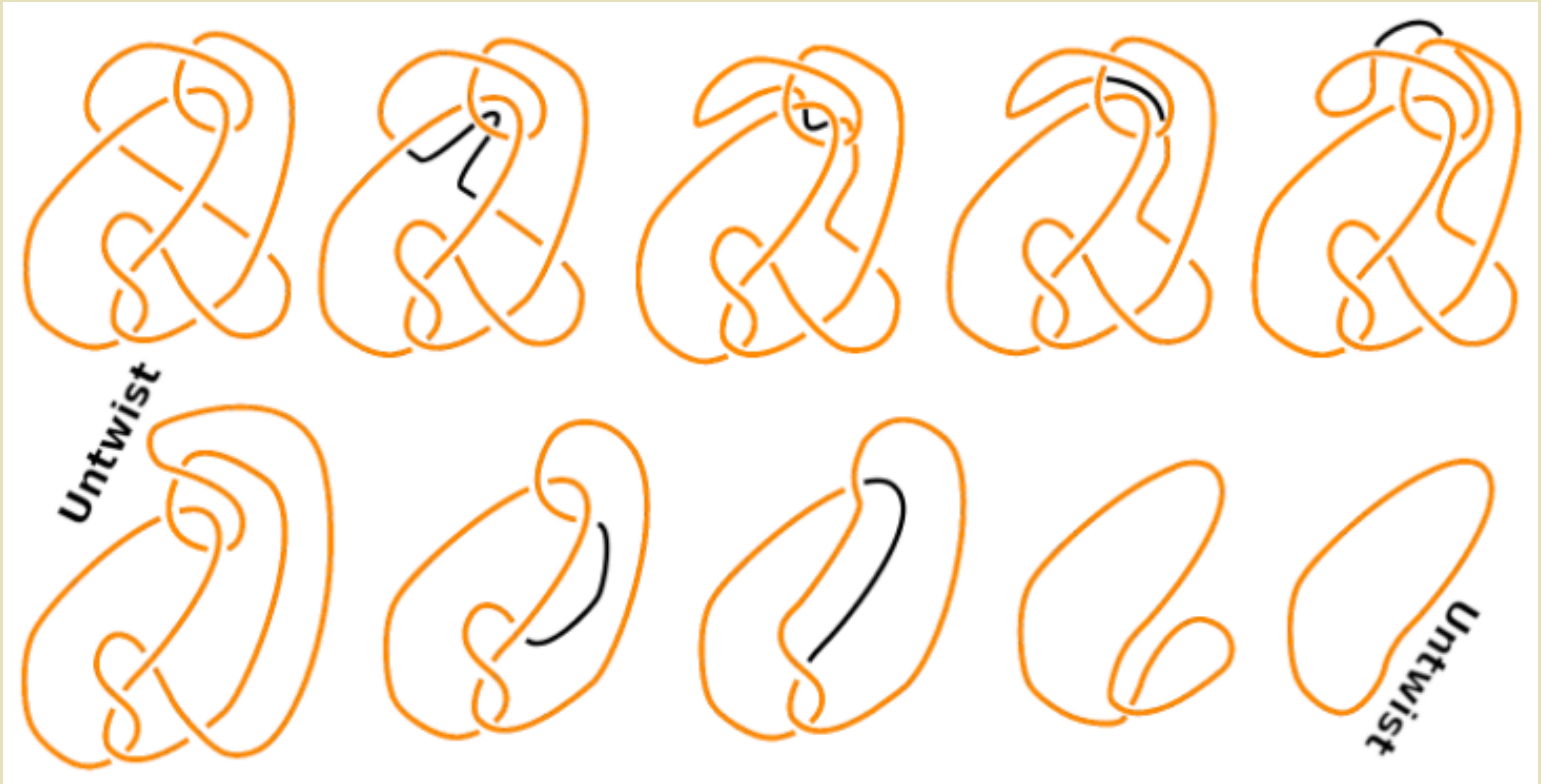
The pipeline



- **Step 1** First inflate the diagram in a controlled but nontrivial way
- **Essentially** A first RL machine guesses promising diagrams and crossings
- **Hope** A larger diagram may reveal a much better crossing to flop

The pipeline

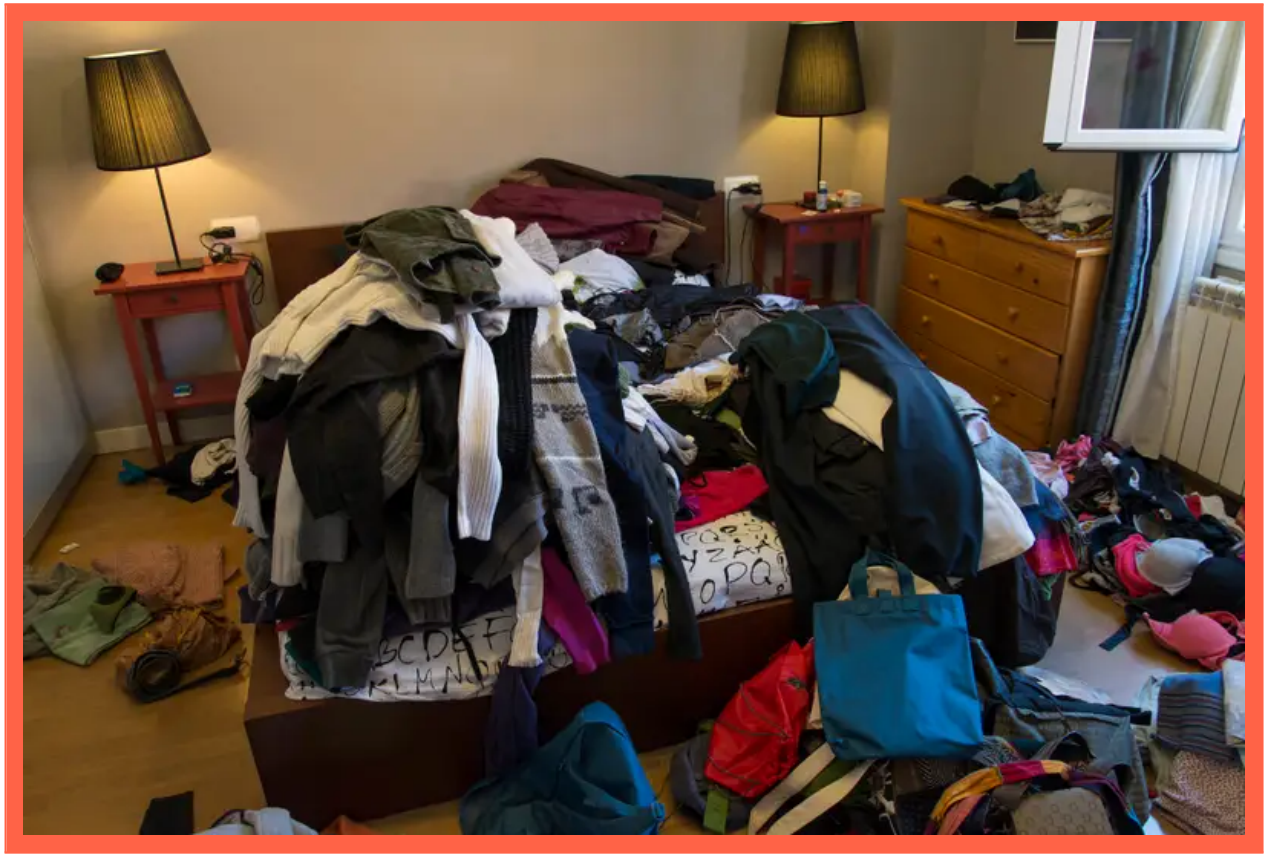
Why inflate first?



In general: sometimes diagrams are "just stuck"
The decisive crossing change may simply not appear on small diagrams
so inflation creates better chances for a good flop

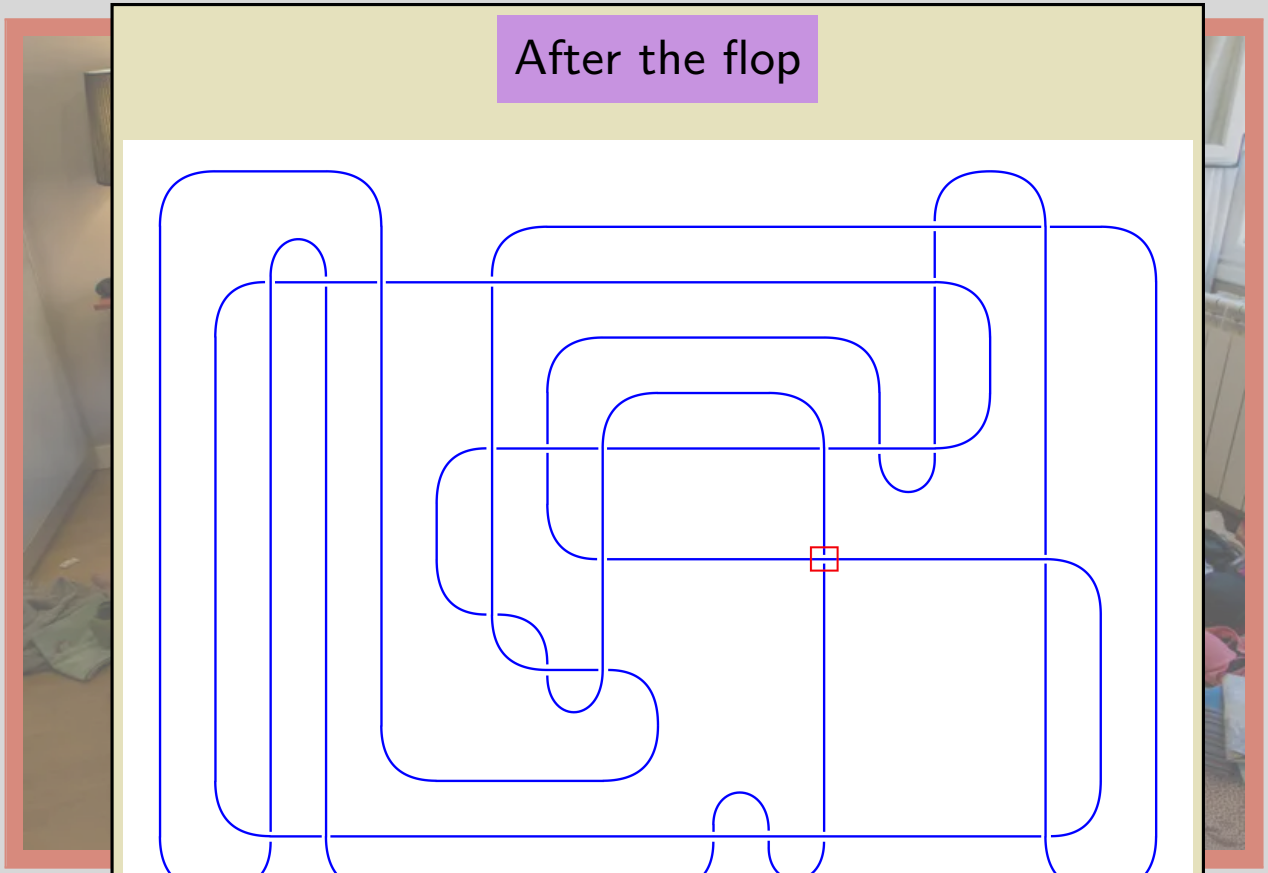
- St
- Essentially A first RE machine guesses promising diagrams and crossings
- **Hope** A larger diagram may reveal a much better crossing to flop

The pipeline



- Step 2 Flop one chosen (RL agent 1) crossing on that inflated diagram
- Best case This lands us dramatically closer to the unknot than before
- Reality The new diagram is usually still much too messy to read

The pipeline

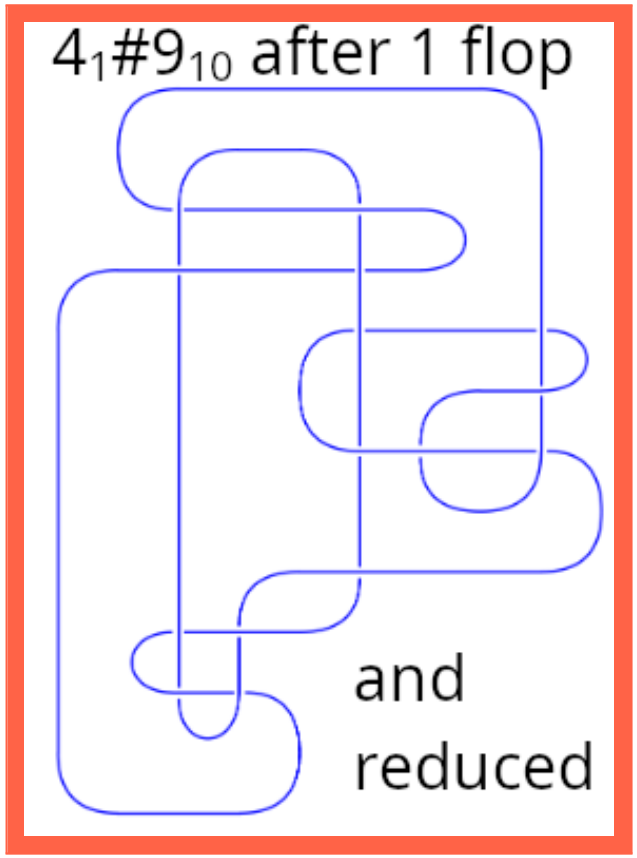


After the flop

Even after the right crossing change
the diagram usually still looks quite hopeless
so we pass it on to a second RL agent

- Step 2 Flop … ted diagram
- Best case T… han before
- Reality The new diagram is usually still much too messy to read

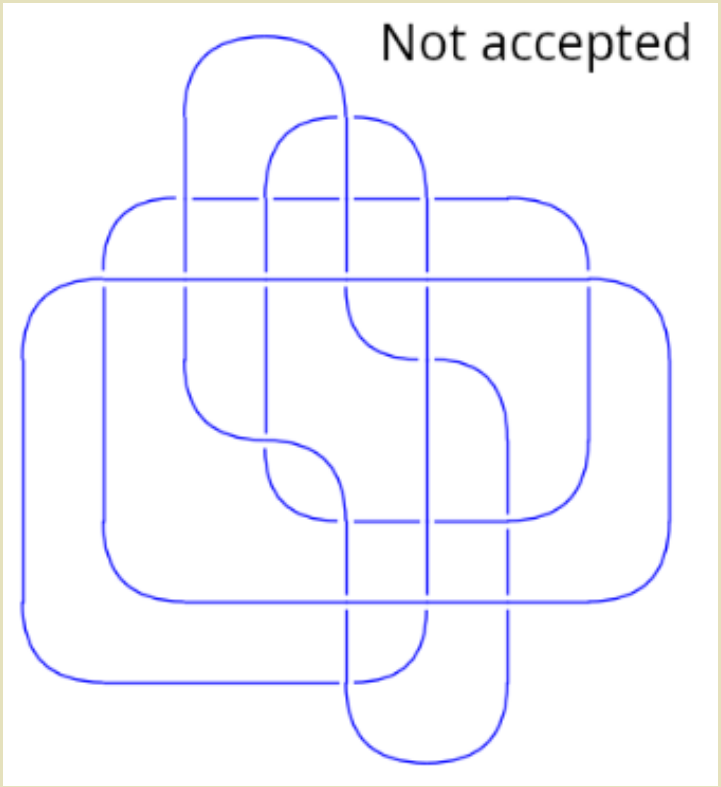
The pipeline



- Step 3 Reduce the result with the unknotter (RL agent 2; more details later)
- Step 4 Then we identify the reduced knot using cheap invariants
- Self-healing Every success produces new training data: the system improves

The pipeline

Crossing reduction is key



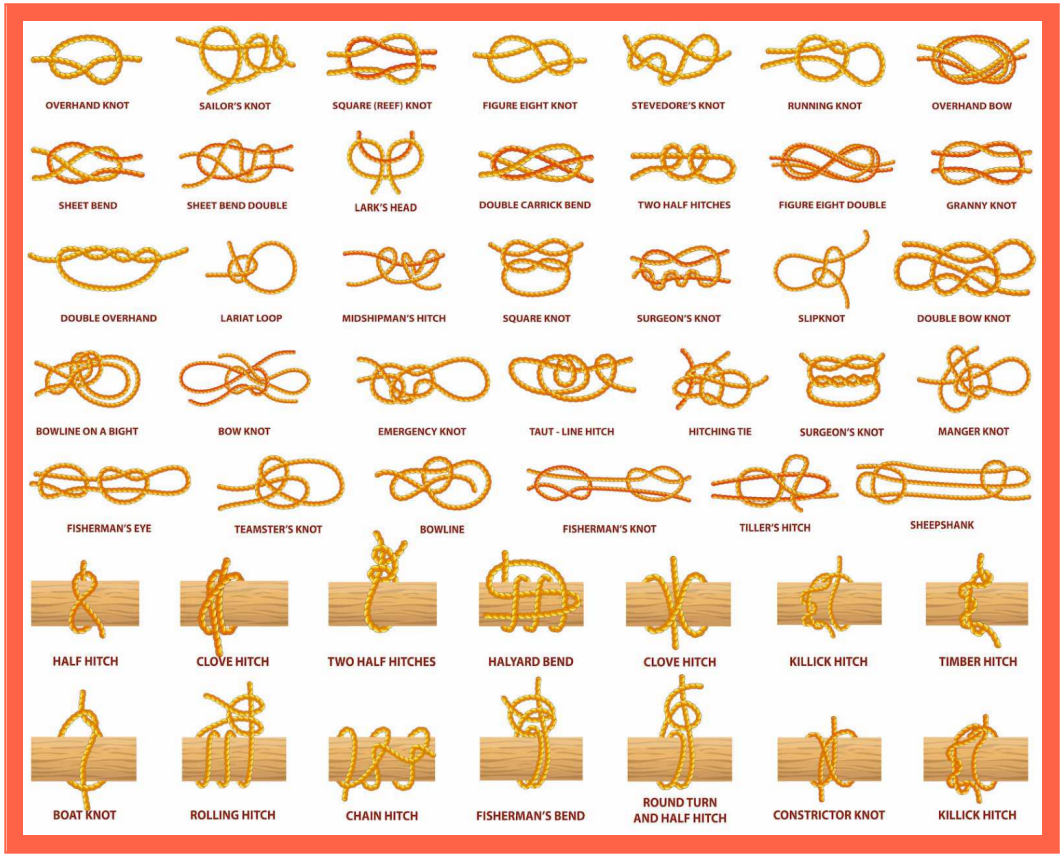
After crossing flop the diagram often still has > 20 crossings
So one can't find it on the list in general (invariants are not perfect!)
so the number of crossings needs to be reduced first

- Ste ails later)
- Ste
- **Self-healing** Every success produces new training data: the system improves

The pipeline

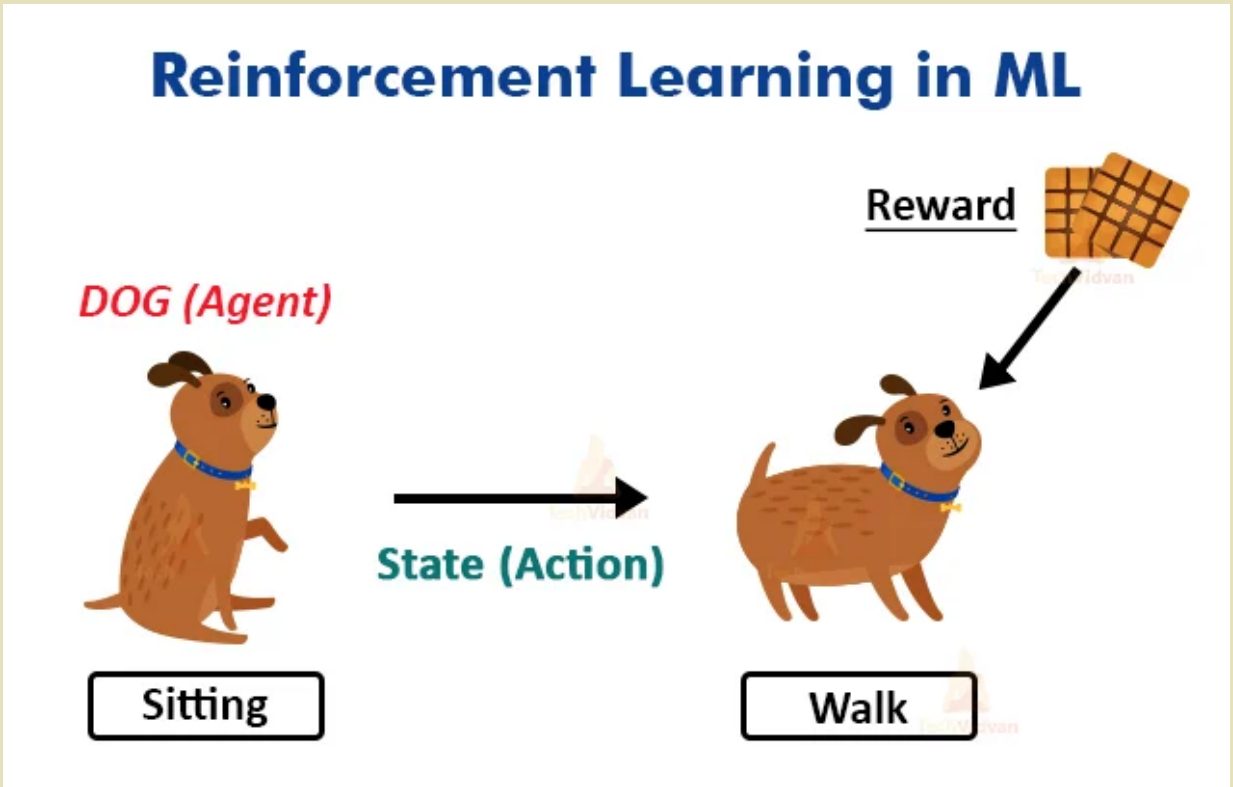
- New 1 Improved bounds for u for prime knots (sometimes identifying u)
- New 2 Verified/new examples of unexpected u under #
- Update soon? It is still running

The unknotter ("find the crossing number")



- RL = learning a strategy from rewards (and penalties)
- Here The environment is the space of knot diagrams and local moves
- Goal Learn which move is most promising from the current diagram

What is RL?



One tries actions in a given environment
and gets feedback on whether they were helpful
so over time one learns a good strategy

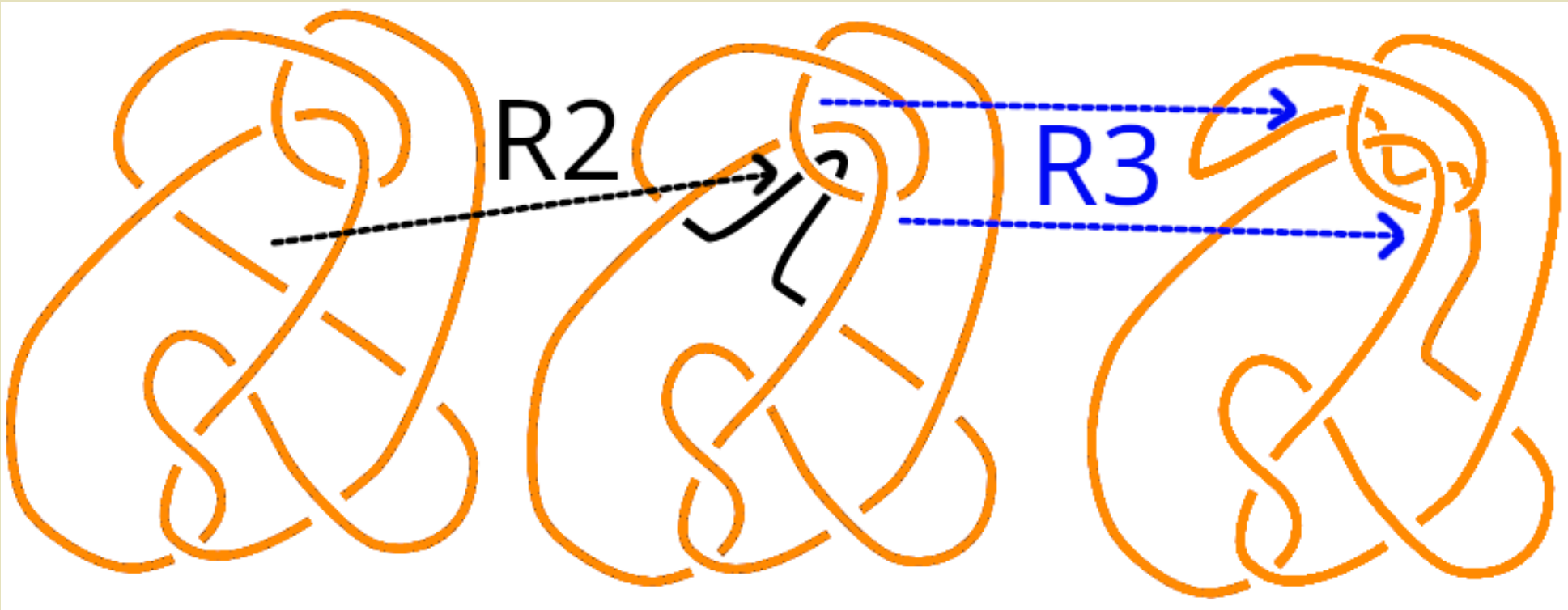
- RL = le
- Here The environment is the space of knot diagrams and local moves
- Goal Learn which move is most promising from the current diagram

The unknotter ("find the crossing number")

- Moves Simplify (R1, R2), shuffle (R3), and also add crossings (R1, R2)
- Reward Simpler diagrams get rewarded but getting stuck is penalized
- Subtlety Good moves do not always look good immediately

The unknotter ("find the crossing number")

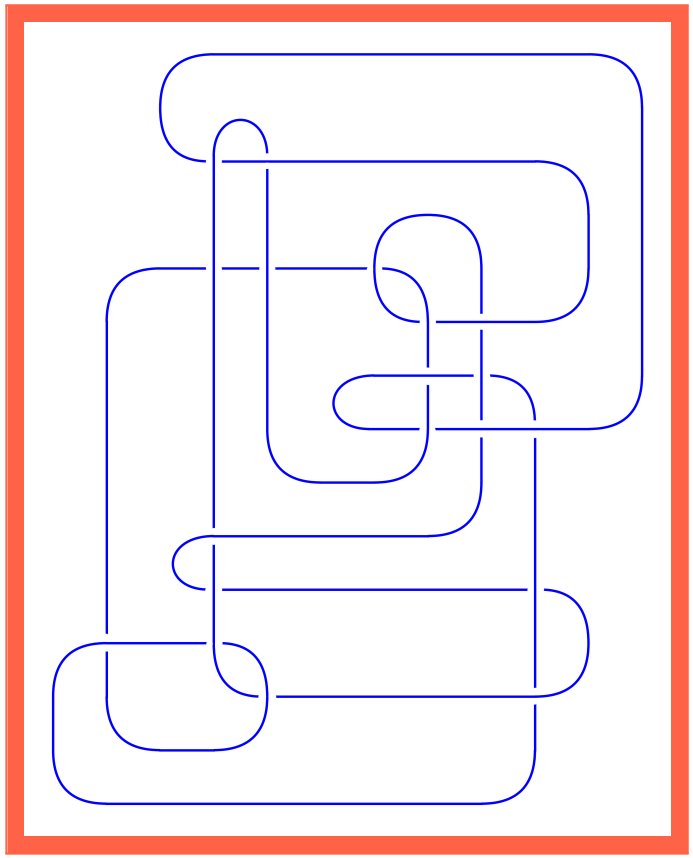
The crucial surprise



If one only simplifies greedily
one may miss the real route completely
so first adding crossings can be the winning move

- **Reward** Simpler diagrams get rewarded but getting stuck is penalized
- **Subtlety** Good moves do not always look good immediately

The unknotter ("find the crossing number")



- Test set There are families of hard and very hard unknot diagrams
- Meaning "No obvious simplification" (see Applebaum et al.)
- Point These are the right benchmarks if one wants to be taken seriously

The unknotter ("find the crossing number")

Hard vs. very hard

From Applebaum et al.:

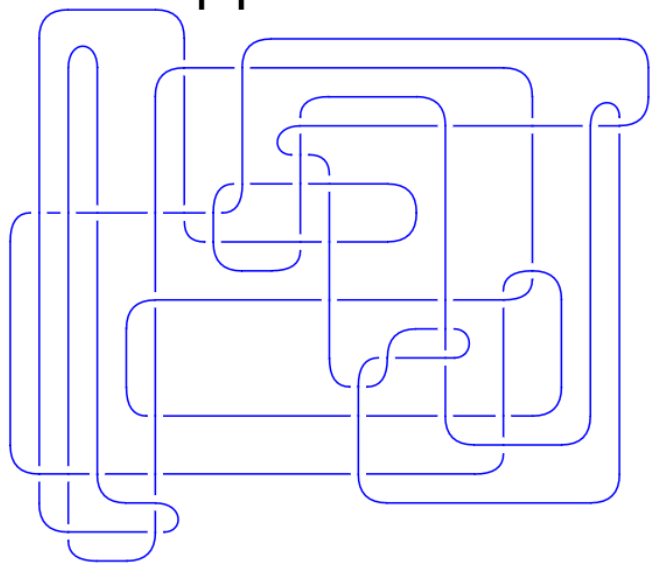


FIGURE 17. A 42-crossing hard unknot diagram with 6225 R3-equivalent diagrams that we have not been able to simplify by calling SnapPy's 'global' heuristic 10000 times.

These examples are designed to defeat short-sighted simplification strategies so they are a very meaningful stress test

- Test set There a … ot diagrams
- Meaning "No o… et al.)
- Point These are the right benchmarks if one wants to be taken seriously

The unknotter ("find the crossing number")

	run	resolved_count	resolved_pct	median_min_crossings	max_min_crossings
0	1	367	95.324675	0.0	46
1	2	367	95.324675	0.0	38
2	3	365	94.805195	0.0	39
3	4	357	92.727273	0.0	46
4	5	357	92.727273	0.0	41
5	6	360	93.506494	0.0	46
6	7	367	95.324675	0.0	39
7	8	368	95.584416	0.0	46
8	9	365	94.805195	0.0	46
9	10	368	95.584416	0.0	40

- Result On these hard examples the unknotter performs surprisingly well
- Strong point It finds genuinely nontrivial reduction sequences in practice
- Message Letting the machine first add crossings was exactly the right idea

The unknotter (“find the crossing number”)

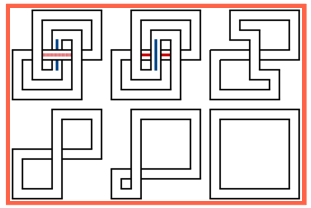
	run	r	ssings
0	1		46
1	2		38
2	3		39
3	4		46
4	5		41
5	6		46
6	7		39
7	8		46
8	9		46
9	10		40

Hard unknot punchline

The unknotter is not merely a toy (no, it is and I like it)
it succeeds on serious benchmark families
and finds genuinely useful reduction paths

- **Result** C singly well
- **Strong p** in practice
- **Message** Letting the machine first add crossings was exactly the right idea

Unknotting is hard



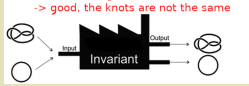
- **Unknotting number u** = minimal number of crossings flops needed to reach the unknot
- **Dream** You can find u on a minimal crossing diagram
- **Problem (why this is so difficult, part a)** That is not true

Unknotting is hard

Problem (why this is so difficult, part b)

The invariant gives the same answer
-> bad, we can't say anything

The invariant gives different answers
-> good, the knots are not the same



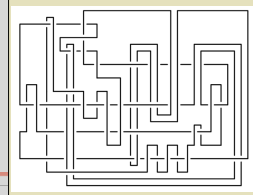
I didn't know any "good" invariant
that gives "interesting" information about u

Side quest With details to come, we can change that!

- **Unknotting** ach the unknot
- **Dream** Y
- **Problem (why this is so difficult, part a)** That is not true

Unknotting is hard

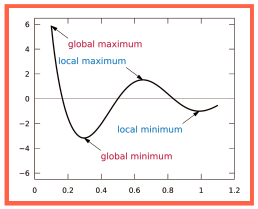
For $7_1\#\mathrm{mir}(7_1)$



The minimal known diagram with $u = 5$
has 56 crossings (**Wang–Zhang 2025**)

- **Open for a l** connected sum #
- **No! Brittenh** real unknotting)
- **Key** What I am going to explain is a variation of their ideas

Unknotting is hard



- **Problem a (rephrased)** Human intuition is local; unknotting often is not
- **Problem b** Friendly invariants have not much to say about u
- **Idea** A perfect task for AI '=' reinforcement learning (RL)

The pipeline

Why start from a list?

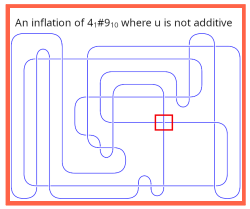
10_1	[-2, 8, 1, -1, 2, -2, >][[2,11,3,12],[4,20,>	1
10_2	[1, 11, 1, -1, 2, -2,>][[1,13,2,12],[3,14,<	3
10_3	[-4, 6, 1, -1, 2, -3, <][[2,10,3,9],[4,17,5>	2
10_4	[-5, 5, 1, -2, 3, -3, <][[2,13,3,14],[4,11,>	2
10_5	[-9, 1, -1, 2, -3, 4, <][[2,13,3,14],[4,17,>	2
10_6	[0, 10, 1, -1, 3, -4,>][[2,11,3,12],[4,16,>][2,3]	
10_7	[-1, 9, 1, -2, 4, -5, >][[1,12,2,13],[3,15,<	1
10_8	[-2, 8, 1, -1, 2, -3, <][[1,12,2,13],[3,10,<	2
10_9	[-3, 7, 1, -2, 3, -4, <][[1,13,2,12],[3,16,<	1
10_10	[-7, 3, -1, 2, -3, 5, <][[1,11,2,10],[3,18,<	1
10_11	[-3, 7, 1, -1, 3, -5, <][[2,15,3,16],[4,13,>][2,3]	
10_12	[-8, 2, -1, 2, -4, 6, <][[2,11,3,12],[4,18,>	2
10_13	[-4, 6, -1, 2, 5, -7, >][[2,17,3,18],[4,8,5,>	2
10_14	[0, 10, 1, -2, 4, -6,>][[1,14,2,15],[3,17,<	2
10_15	[-6, 4, -1, 2, -4, 6, <][[2,6,3,5],[4,16,5,>	2
10_16	[-3, 7, 1, -2, 4, -5, >][[1,10,2,11],[3,15,<	2
10_17	[-5, 5, -1, 2, -3, 5, <][[6,2,7,1],[12,4,13,>	1

One targets knots with a known lower bound
but still no sharp upper bound in hand
so every success gives genuinely new information

- **Input** Take vn)
- **Goal** Impro the unknot
- **Key** Do not insist on working with the smallest visible diagram

The pipeline

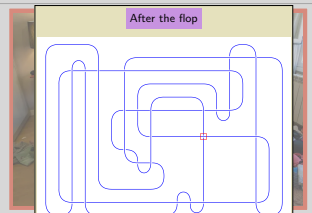
An inflation of $4_1\#9_{10}$ where u is not additive



- **Step 1** First inflate the diagram in a controlled but nontrivial way
- **Essentially** A first RL machine guesses promising diagrams and crossings
- **Hope** A larger diagram may reveal a much better crossing to flop

The pipeline

After the flop

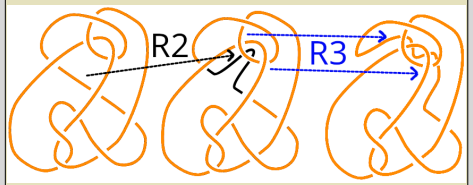


Even after the right crossing change
the diagram usually still looks quite hopeless
so we pass it on to a second RL agent

- **Step 2** Flop ted diagram
- **Best case** T han before
- **Reality** The new diagram is usually still much too messy to read

The unknotter ("find the crossing number")

The crucial surprise



If one only simplifies greedily
one may miss the real route completely
so first adding crossings can be the winning move

- **Reward** Simpler diagrams get rewarded but getting stuck is penalized
- **Subtlety** Good moves do not always look good immediately

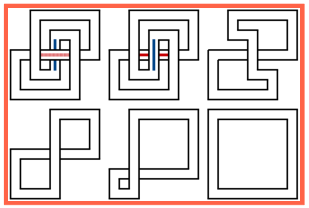
The unknotter ("find the crossing number")

	run	resolved_count	resolved_pct	median_min_crossings	max_min_crossings
0	1	367	95.324675	0.0	46
1	2	367	95.324675	0.0	38
2	3	365	94.805195	0.0	39
3	4	357	92.727273	0.0	46
4	5	357	92.727273	0.0	41
5	6	360	93.506494	0.0	46
6	7	367	95.324675	0.0	39
7	8	368	95.584416	0.0	46
8	9	365	94.805195	0.0	46
9	10	368	95.584416	0.0	40

- **Result** On these hard examples the unknotter performs surprisingly well
- **Strong point** It finds genuinely nontrivial reduction sequences in practice
- **Message** Letting the machine first add crossings was exactly the right idea

There is still much to do...

Unknotting is hard



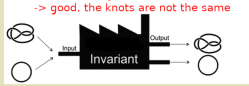
- **Unknotting number u** = minimal number of crossings flops needed to reach the unknot
- **Dream** You can find u on a minimal crossing diagram
- **Problem (why this is so difficult, part a)** That is not true

Unknotting is hard

Problem (why this is so difficult, part b)

The invariant gives the same answer
-> bad, we can't say anything

The invariant gives different answers
-> good, the knots are not the same



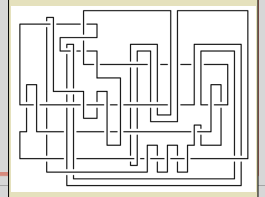
I didn't know any "good" invariant
that gives "interesting" information about u

Side quest With details to come, we can change that!

- **Unknotting** ach the unknot
- **Dream** Y
- **Problem (why this is so difficult, part a)** That is not true

Unknotting is hard

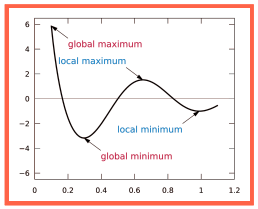
For $7_1\#\mathrm{mir}(7_1)$



The minimal known diagram with $u = 5$
has 56 crossings (**Wang–Zhang 2025**)

- **Open for a l** connected sum #
- **No! Brittenh** real unknotting)
- **Key** What I am going to explain is a variation of their ideas

Unknotting is hard



- **Problem a (rephrased)** Human intuition is local; unknotting often is not
- **Problem b** Friendly invariants have not much to say about u
- **Idea** A perfect task for AI '=' reinforcement learning (RL)

The pipeline

Why start from a list?

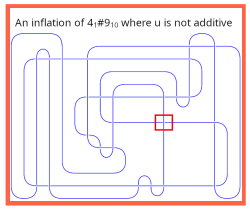
10_1	[-2, 8, 1, -1, 2, -2, >][[2,11,3,12],[4,20,>]	1
10_2	[1, 11, 1, -1, 2, -2,>][[1,13,2,12],[3,14,<	3
10_3	[-4, 6, 1, -1, 2, -3, <][[2,10,3,9],[4,17,5>	2
10_4	[-5, 5, 1, -2, 3, -3, <][[2,13,3,14],[4,11,>	2
10_5	[-9, 1, -1, 2, -3, 4, <][[2,13,3,14],[4,17,>	2
10_6	[0, 10, 1, -1, 3, -4,>][[2,11,3,12],[4,16,>][2,3]	
10_7	[-1, 9, 1, -2, 4, -5, >][[1,12,2,13],[3,15,<	1
10_8	[-2, 8, 1, -1, 2, -3, <][[1,12,2,13],[3,10,<	2
10_9	[-3, 7, 1, -2, 3, -4, <][[1,13,2,12],[3,16,<	1
10_10	[-7, 3, -1, 2, -3, 5, <][[1,11,2,10],[3,18,<	1
10_11	[-3, 7, 1, -1, 3, -5, <][[2,15,3,16],[4,13,>][2,3]	
10_12	[-8, 2, -1, 2, -4, 6, <][[2,11,3,12],[4,18,>	2
10_13	[-4, 6, 1, -2, 5, -7, >][[2,17,3,18],[4,8,5>	2
10_14	[0, 10, 1, -2, 4, -6,>][[1,14,2,15],[3,17,<	2
10_15	[-6, 4, -1, 2, -4, 6, >][[2,6,3,5],[4,16,5,3>	2
10_16	[-3, 7, 1, -2, 4, -5, >][[1,10,2,11],[3,15,<	2
10_17	[-5, 5, -1, 2, -3, 5, >][[6,2,7,1],[12,4,13,>	1

One targets knots with a known lower bound
but still no sharp upper bound in hand
so every success gives genuinely new information

- **Input** Take vn)
- **Goal** Impro the unknot
- **Key** Do not insist on working with the smallest visible diagram

The pipeline

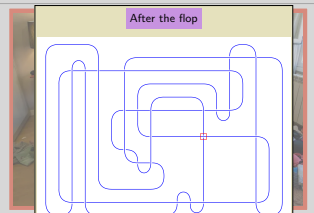
An inflation of $4_1\#9_{10}$ where u is not additive



- **Step 1** First inflate the diagram in a controlled but nontrivial way
- **Essentially** A first RL machine guesses promising diagrams and crossings
- **Hope** A larger diagram may reveal a much better crossing to flop

The pipeline

After the flop

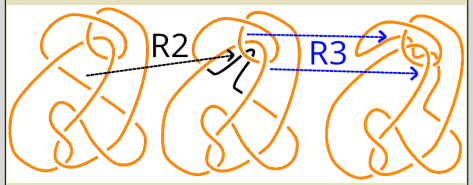


Even after the right crossing change
the diagram usually still looks quite hopeless
so we pass it on to a second RL agent

- **Step 2** Flop ted diagram
- **Best case** T han before
- **Reality** The new diagram is usually still much too messy to read

The unknotter ("find the crossing number")

The crucial surprise



If one only simplifies greedily
one may miss the real route completely
so first adding crossings can be the winning move

- **Reward** Simpler diagrams get rewarded but getting stuck is penalized
- **Subtlety** Good moves do not always look good immediately

The unknotter ("find the crossing number")

	run	resolved_count	resolved_pct	median_min_crossings	max_min_crossings
0	1	367	95.324675	0.0	46
1	2	367	95.324675	0.0	38
2	3	365	94.805195	0.0	39
3	4	357	92.727273	0.0	46
4	5	357	92.727273	0.0	41
5	6	360	93.506494	0.0	46
6	7	367	95.324675	0.0	39
7	8	368	95.584416	0.0	46
8	9	365	94.805195	0.0	46
9	10	368	95.584416	0.0	40

- **Result** On these hard examples the unknotter performs surprisingly well
- **Strong point** It finds genuinely nontrivial reduction sequences in practice
- **Message** Letting the machine first add crossings was exactly the right idea

Thanks for your attention!