Computational Complexity and Knot Theory

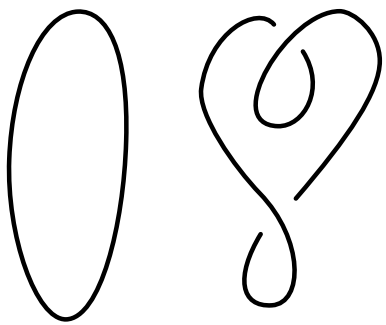
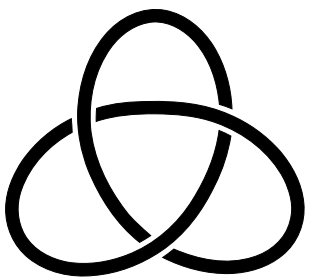
Arnaud de Mesmay (CNRS, LIGM, Université Gustave Eiffel, Paris)

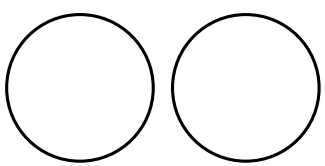
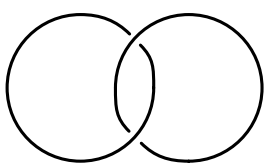
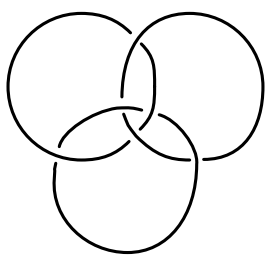
AMS-SMF-EMS Meeting, Grenoble, 2022

Knot Theory

Knots

- A *knot* is a nice map $K : S^1 \to \mathbb{R}^3$.
- Two knots are *equivalent* if they are *ambient isotopic*, i.e., if there exists a continuous deformation from one into the other without crossings.
- A *link* is a disjoint union of knots.

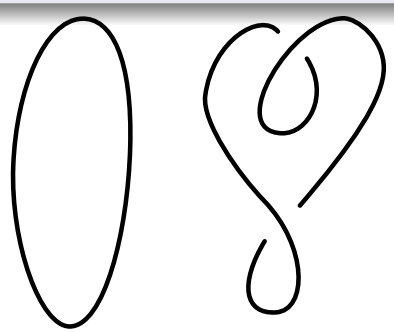
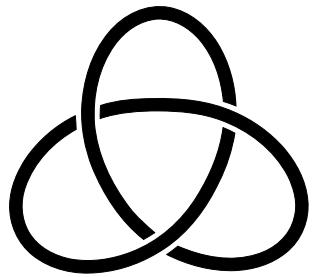



Knot diagrams

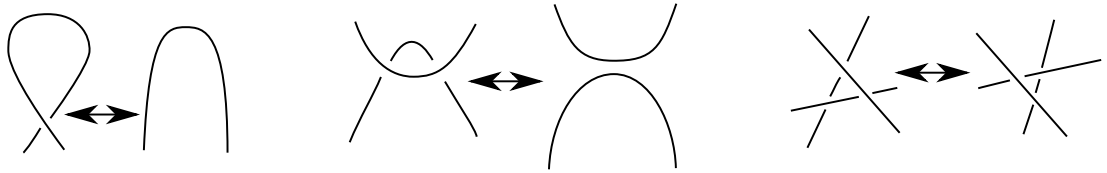
Diagrams

- A *knot diagram* is a 2D-projection of a knot where at every vertex, one indicates which strand goes above and below.
- The *crossing number* of a knot K is a minimum number of crossings over all knot diagrams for K.


Theorem (Reidemeister)

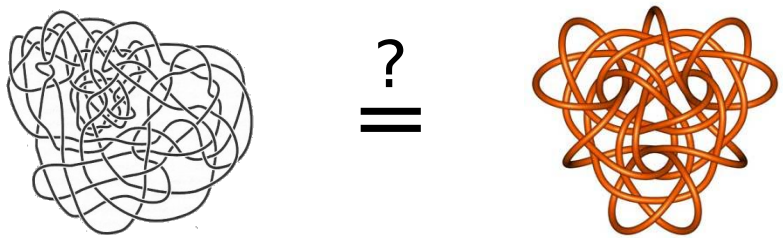
Two knot diagrams correspond to equivalent knots if and only if they can be related by a sequence of Reidemeister moves.



Knot equivalence

Knot equivalence

Input: Two knots K_1 and K_2 represented by diagrams.
Output: Is K_1 equivalent to K_2?

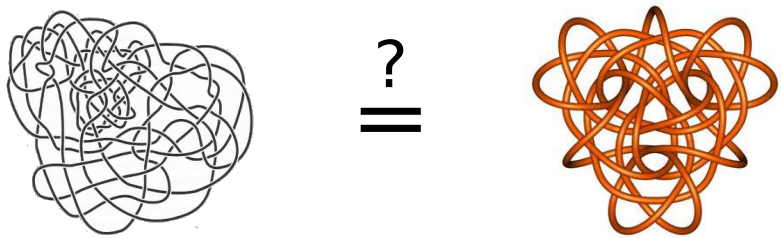


What is the best known algorithm for this problem?

Knot equivalence

Knot equivalence

Input: Two knots K_1 and K_2 represented by diagrams.
Output: Is K_1 equivalent to K_2?



What is the best known algorithm for this problem?

- Decidable [Haken'68, Hemion '79, Matveev '07].
- Best bound on Reidemeister moves is from [Lackenby-Coward '14]:

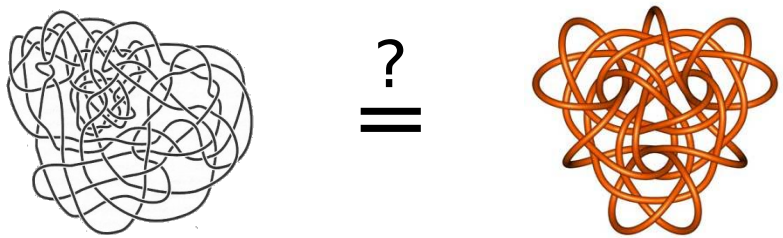
$$2^{2^{\cdot^{\cdot^{2^{n_1+n_2}}}}} \Big\} \text{ height } c^{n_1+n_2} \text{ where } c = 10^{1000000}.$$

- [Kuperberg '19] provides an elementary algorithm, i.e., with a tower of exponentials of constant size.

Knot equivalence

Knot equivalence

Input: Two knots K_1 and K_2 represented by diagrams.
Output: Is K_1 equivalent to K_2?



What is the best known algorithm for this problem?

- Decidable [Haken'68, Hemion '79, Matveev '07].
- Best bound on Reidemeister moves is from [Lackenby-Coward '14]:

$$2^{2^{\cdot^{\cdot^{2^{n_1+n_2}}}}} \Big\} \text{ height } c^{n_1+n_2} \text{ where } c = 10^{1000000}.$$

- [Kuperberg '19] provides an elementary algorithm, i.e., with a tower of exponentials of constant size.

This problem does not look easy. Perhaps one can prove that it is hard?

The very basics of computational complexity I

- A decision problem is in **P** (polynomial-time) if there exists an algorithm solving instances of size n in time $p(n)$, where p is a polynomial.
- A decision problem is in **NP** (non-deterministic polynomial-time) if there exists an algorithm *verifying* positive instances of size n *with a hint of size* $h(n)$ in time $p(n)$, where p and h are polynomial.

Example: Non-primality

One can easily verify that a number is non-prime when one is given its prime factors as a hint.

- A decision problem is in co-**NP** if its complement is in **NP**.

Example: Primality

Primality testing is in co-**NP** because of the previous example.

The very basics of computational complexity II

Standard conjectures

P $\neq$ **NP** $\neq$ co-**NP**.

- A problem is **NP**-hard if any problem in **NP** reduces in polynomial time to it. In particular, a problem being both in **P** (resp. co-**NP**) and **NP**-hard would mean that **P** = **NP** (resp. co-**NP**=**NP**).
- Rule of thumb in many parts of theoretical computer science (e.g., graph theory): every reasonable problem, except a few well-known exceptions, is in **P** or is **NP**-hard (see for example the Feder-Vardi conjecture [Bulatov, Zhuk '17])
- Most problems in knot theory are not known to fit in this dichotomy.

NP-hard problems in knot theory

- Is knot equivalence **NP**-hard?

NP-hard problems in knot theory

- Is knot equivalence **NP**-hard? We do not know.
- It is consistent with the state of the art that it can be solved in linear time.

NP-hard problems in knot theory

- Is knot equivalence **NP**-hard? We do not know.
- It is consistent with the state of the art that it can be solved in linear time.
- Most problems in knot theory are very hard to compute in practice. Is any of them **NP**-hard?
- When I started working on these problems ten years ago, there were to my knowledge only two known computational hardness results in knot theory:
  1. Given a knot K in a 3-manifold M, is the genus of K at most g? [Agol-Hass-Thurston '95]
  2. It is $\#P$-hard to compute [Jaeger, Vertigan, Welsh '90], or even approximate [Kuperberg '15] the Jones polynomial of a knot.

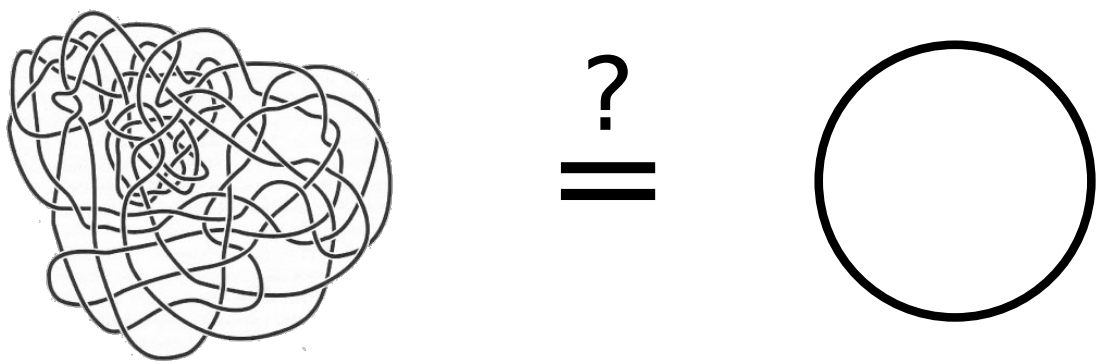
Since then, many new results, but still many open problems.

Why should we care?

1. **NP**-hardness traces a line between problems that can be solved in polynomial time and those that cannot.
2. Some problems in knot theory are very likely not **NP**-hard:

Unknot recognition

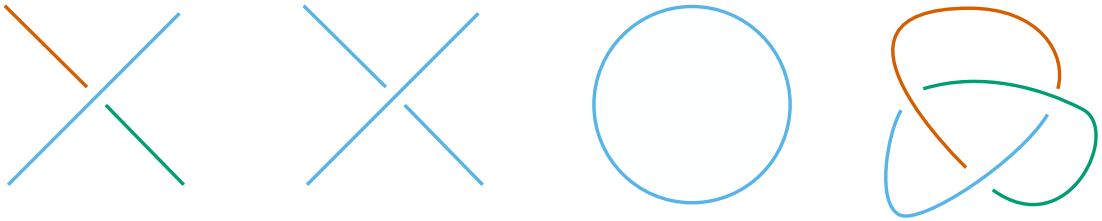
Input: A knot K represented by a diagram.
Output: Is K equivalent to the trivial knot?



- In $\mathbf{NP} \cap co - \mathbf{NP}$ [Hass-Lagarias-Pippenger '99], [Agol'02 $\rightarrow$ Lackenby '18].
- Actually, computing the genus is in $\mathbf{NP} \cap co - \mathbf{NP}$ [Lackenby '21], even in a fixed 3-manifold [Lackenby-Yazdi '20]

Some knot invariants

- Tri-colorability: Can I color my knot using three different colors and the following rules?



- More generally: does there exist a non-trivial homomorphism of $\pi_1(\mathbb{S}^3 \setminus K)$ into a fixed finite group G that sends a meridian to a fixed conjugacy class c?

Theorem (Kuperberg-Samperton '21)

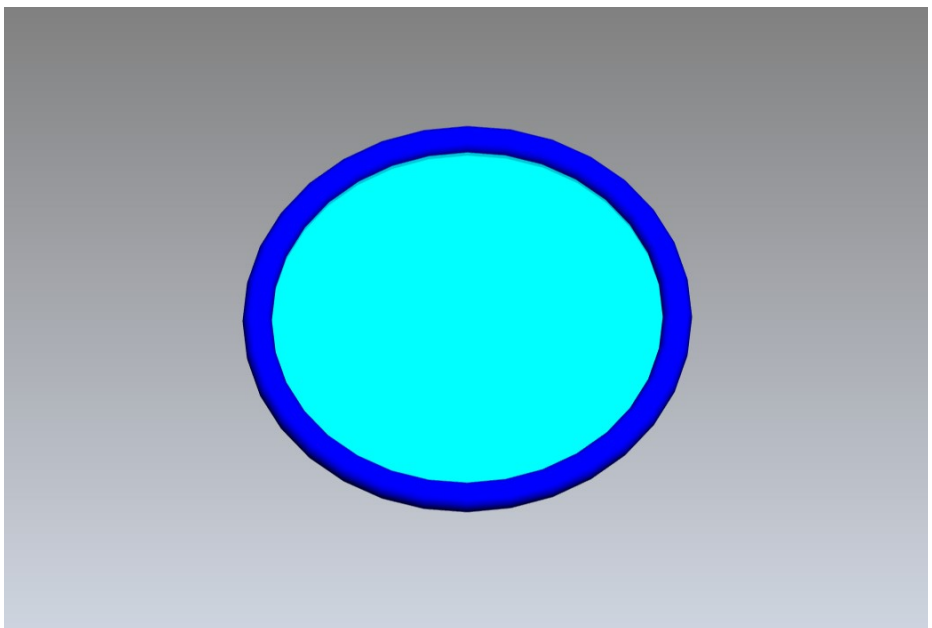
*If G is fixed, non-abelian and simple (for example $\mathcal{A}_5$), this problem is **NP**-hard. (More generally counting the number of representations is $\#P$-hard.)*

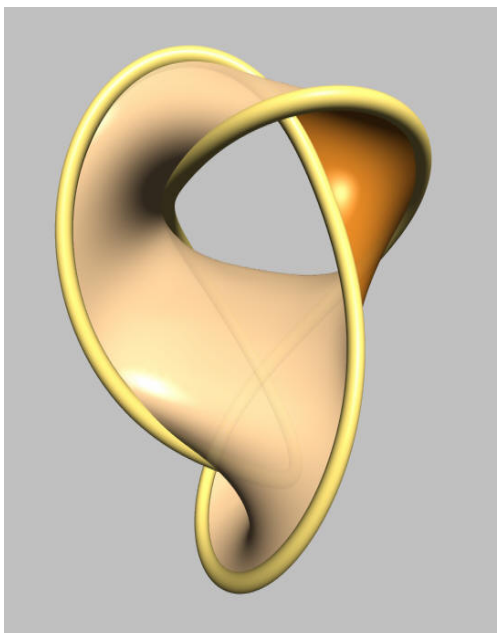
- Polynomial invariants: Computing the Alexander polynomial of a knot can be done in *polynomial-time*, but computing/approximating the Jones polynomial is $\#P$-hard [Jaeger-Vertigan-Welsh '90, Kuperberg '15].

These are the only* known hard problems for *classical knots*. Other hardness results work with *knots in 3-manifolds*, or with *links*.

Some previous NP-hardness results

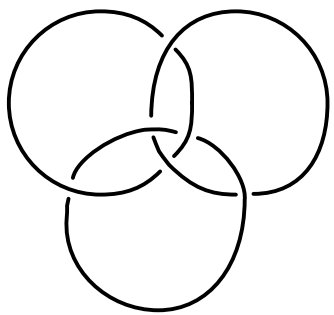
- Knot genus in a 3-manifold [Agol-Hass-Thurston '05].
- Thurston norm of a link [Lackenby '17].





Some previous NP-hardness results

In [dM-Rieck-Sedgwick-Tancer '21], we started from a nice similarity between Borromean rings and SAT clauses to show that the following problems are NP-hard:

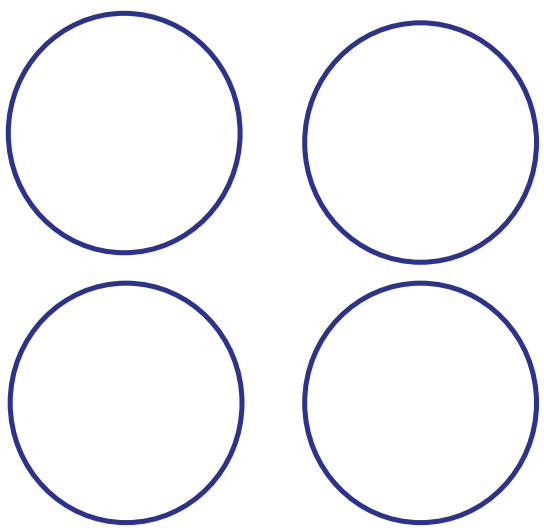


$$C = v_1 \vee v_2 \vee v_3$$

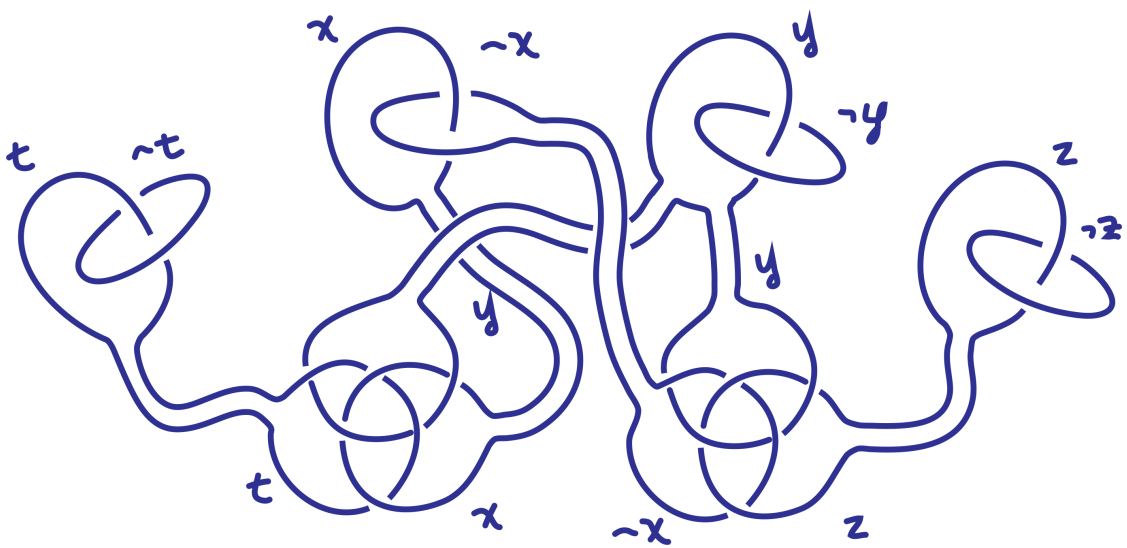
Some previous NP-hardness results

- Finding a sublink [Lackenby '17], finding a trivial sublink [Koenig-Tsvietkova'21],[dM-Rieck-Sedgwick-Tancer '21].

$$\phi = (t \vee x \vee y) \wedge (\neg x \vee y \vee z)$$

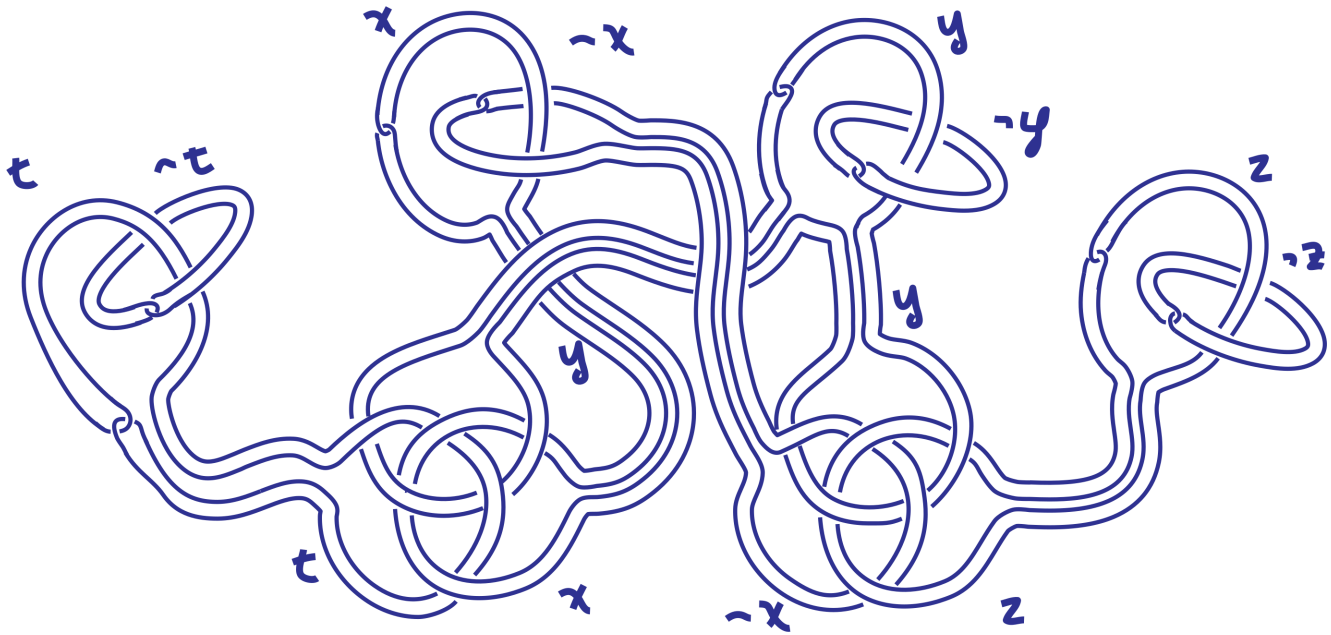


$\subset$



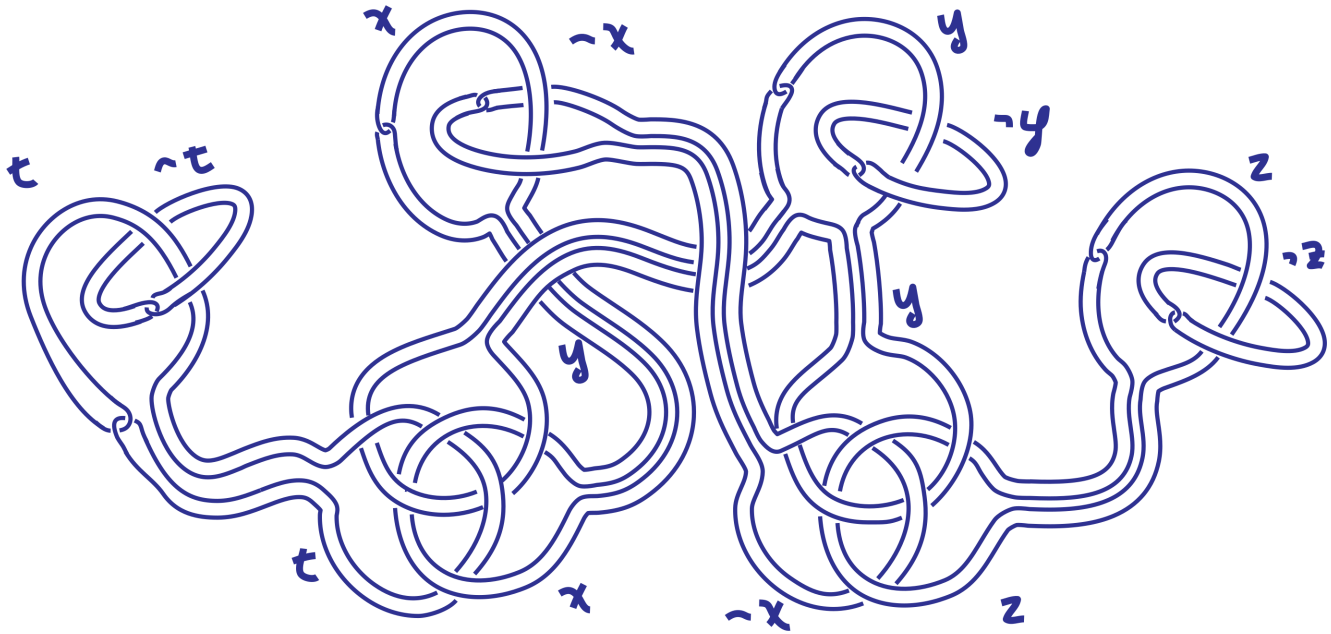
Some previous NP-hardness results

- Unlinking number [Koenig-Tsvietkova'21],[dM-Rieck-Sedgwick-Tancer'21].



Some previous NP-hardness results

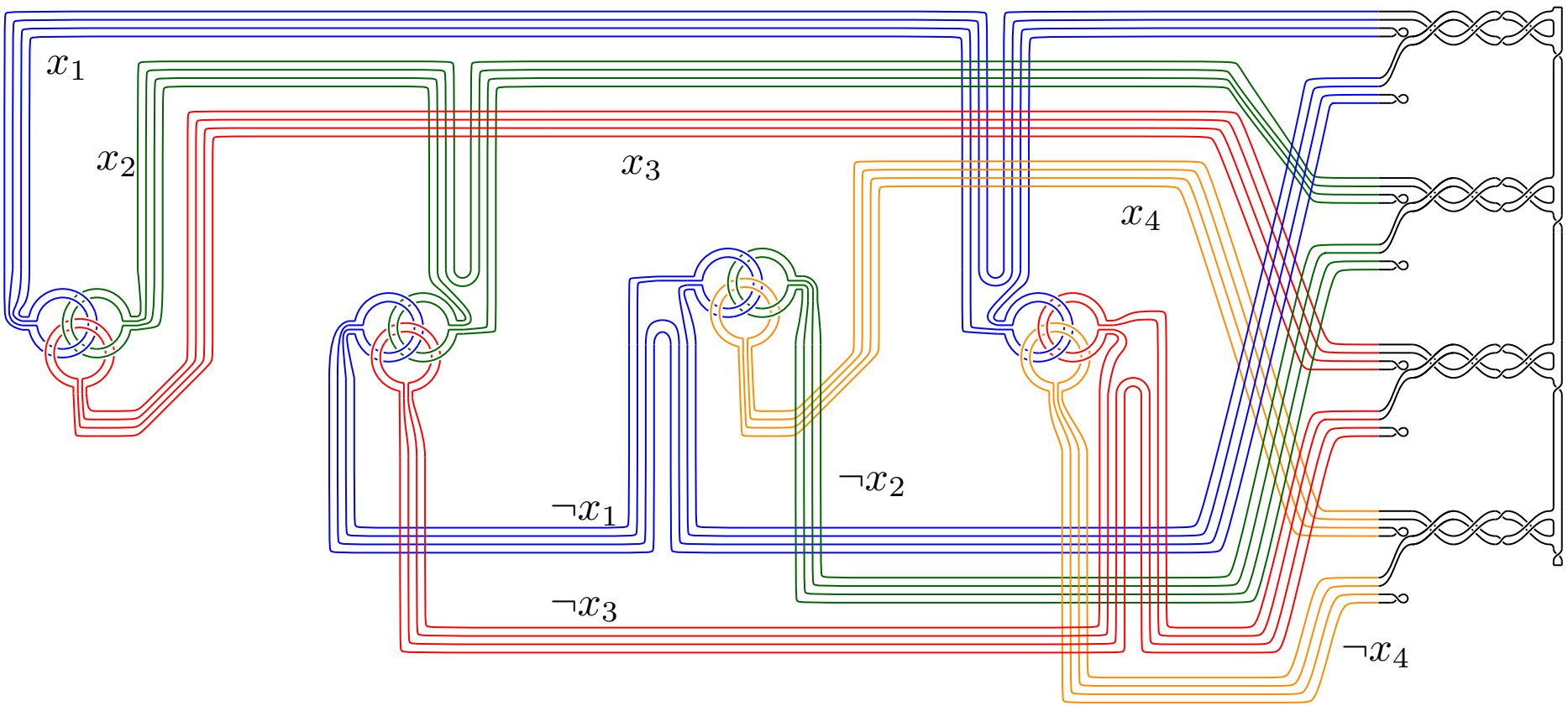
- Four-ball Euler characteristic $\chi_4(L)$ [dM-Rieck-Sedgwick-Tancer'21].



Some previous NP-hardness results

- Deciding whether a *knot* can be turned into a trivial diagram using at most k Reidemeister moves [dM-Rieck-Sedgwick-Tancer '21].

$$\Phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4)$$



Computing the crossing number

Crossing number

Input: A knot/link diagram D.
Output: Is the crossing number of the knot/link at most k?

Best known algorithm to decide whether the crossing number of a link L is at most k:

for $i = 1 \ldots k$ **do**
    for All the link diagrams D with i crossings **do**
        Test whether D and L are the same link.
    end for
end for

Marc Lackenby

"[This algorithm] is obviously not very efficient but it seems unlikely that there is any quicker way of determining a link's crossing number in general."

Hardness of the crossing number

Theorem (Schaefer, Sedgwick, dM'20)

*The crossing number problem for links is **NP**-hard.*

Hardness of the crossing number

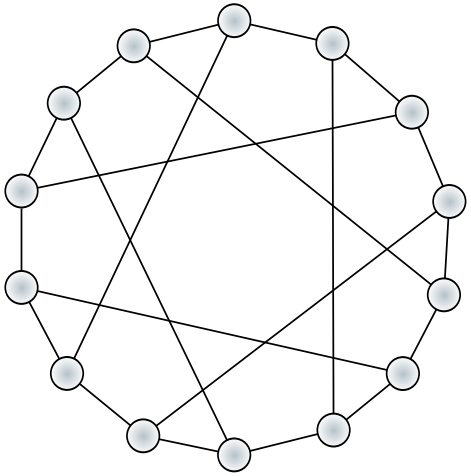
Theorem (Schaefer, Sedgwick, dM'20)

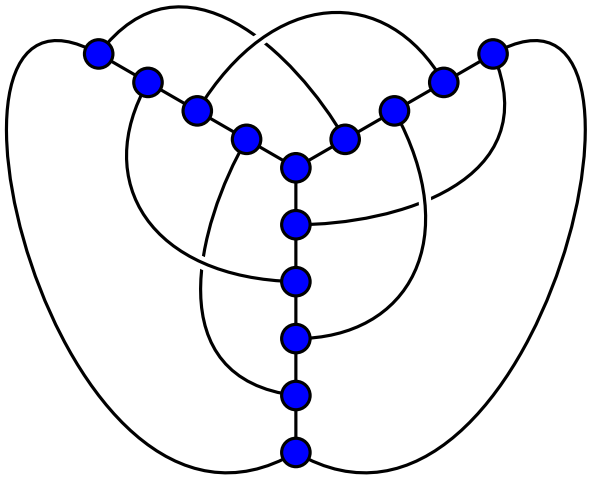
*The crossing number problem for links is **NP**-hard.*

- First reaction: of course it's **NP**-hard, there should be an easy reduction from the *graph crossing number*.
- Yes but not that easy: it is open whether the crossing number of a *knot* is **NP**-hard.

A naive reduction

- The *crossing number* of a graph G is the minimum number of edge crossings in a plane drawing of G.





- It is notoriously unwieldy, for example the exact values of the crossing numbers of complete graphs K_n and complete bipartite graph $K_{m,n}$ are unknown.

Theorem (Garey-Johnson '83)

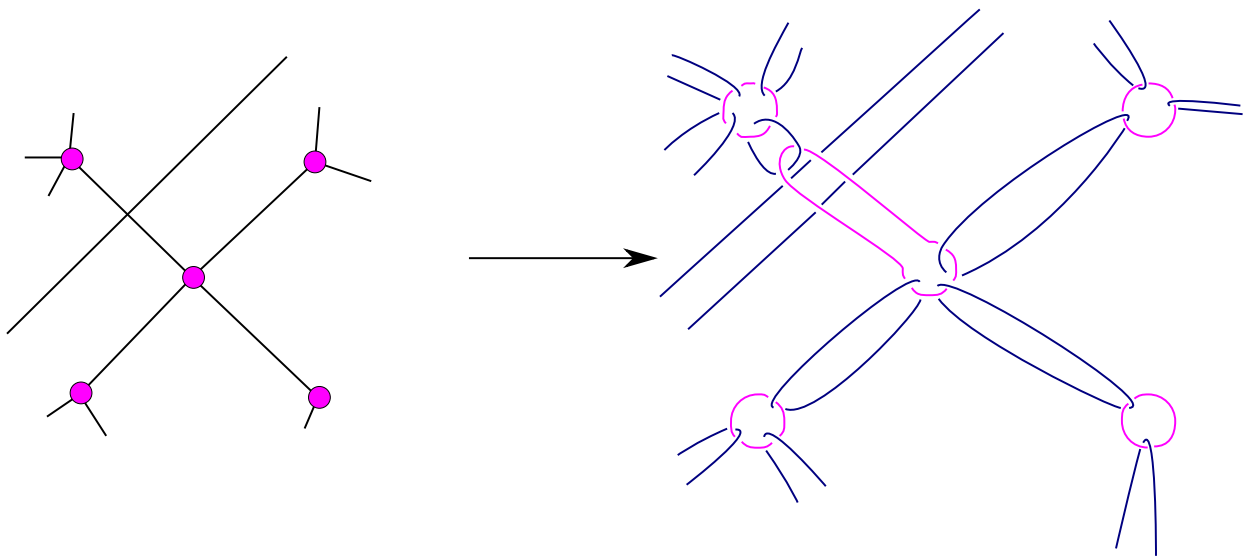
Computing the crossing number of a graph is ***NP****-hard.*

A naive reduction

Theorem (Garey-Johnson '83)

*Computing the crossing number of a graph is **NP**-hard.*

- Transforming a graph into a link.



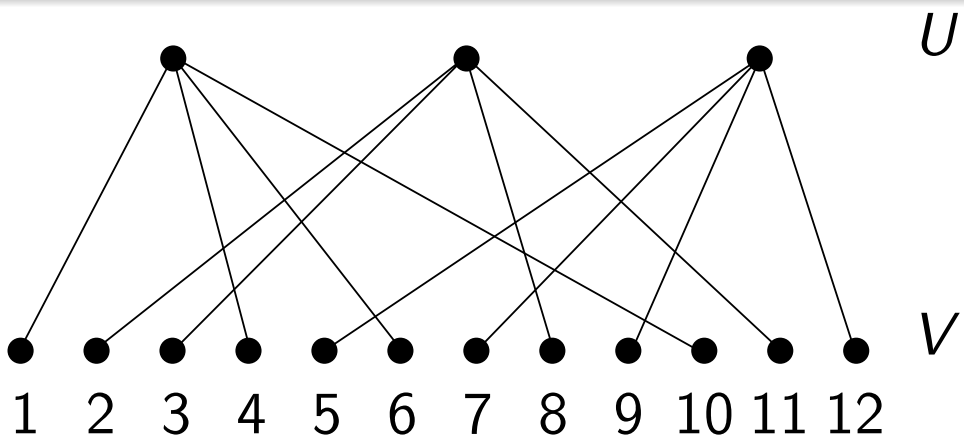
- But when changing the cyclic ordering around the vertex, the link gets all tangled up.
- We must prevent the components corresponding to vertices from stretching.

The actual reduction

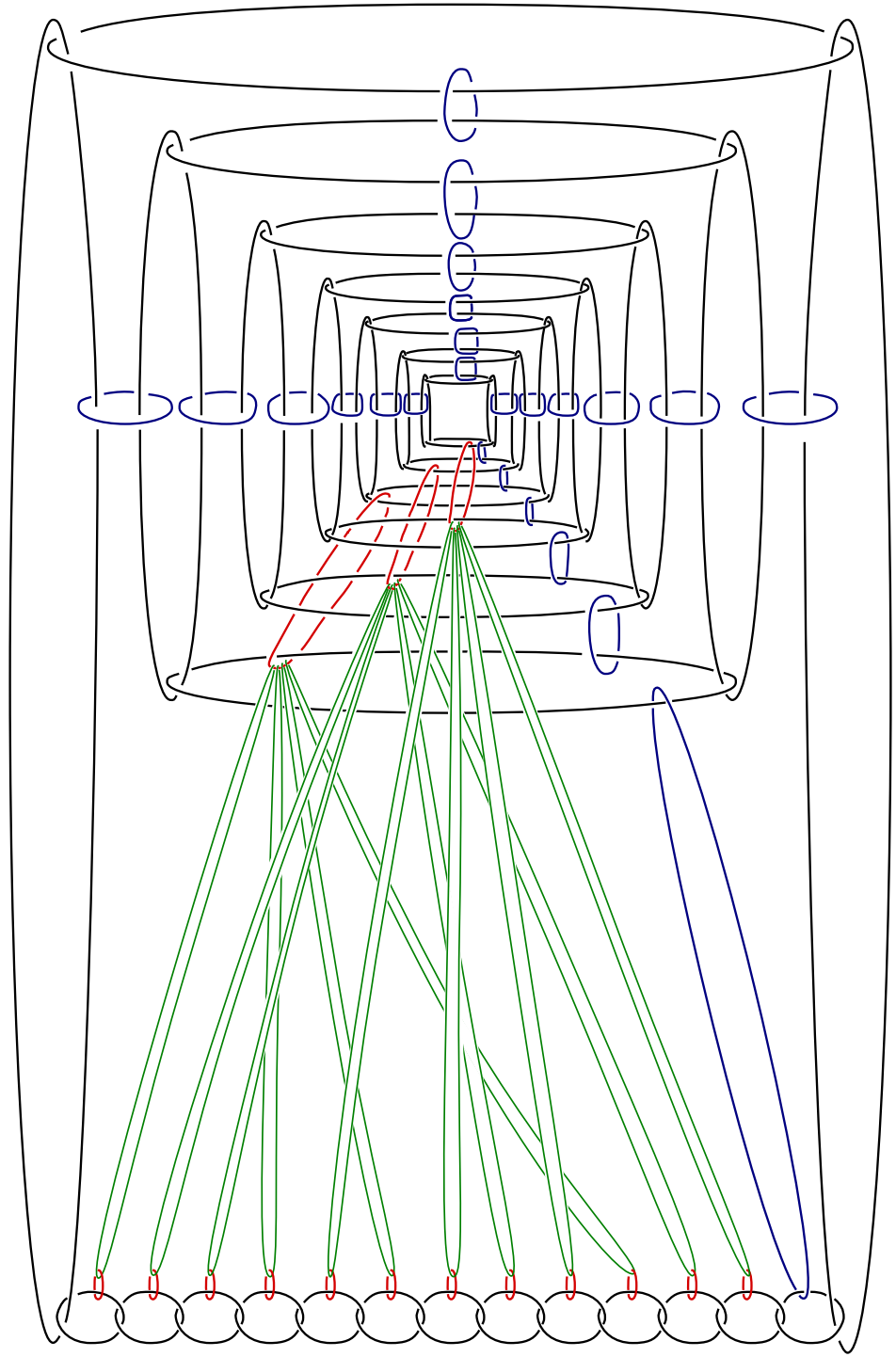
We reduce from a specific variant of the graph crossing number, where the cyclic orderings are fixed:

Theorem (Muñoz-Unger-Vrťo '02)

*Determining the bipartite crossing number of a bipartite graph $G = (U \cup V, E)$ in which all vertices in U have degree 4, all vertices in V have degree 1, and the order of the V-vertices along their line is fixed, is **NP**-complete.*



which we transform into...

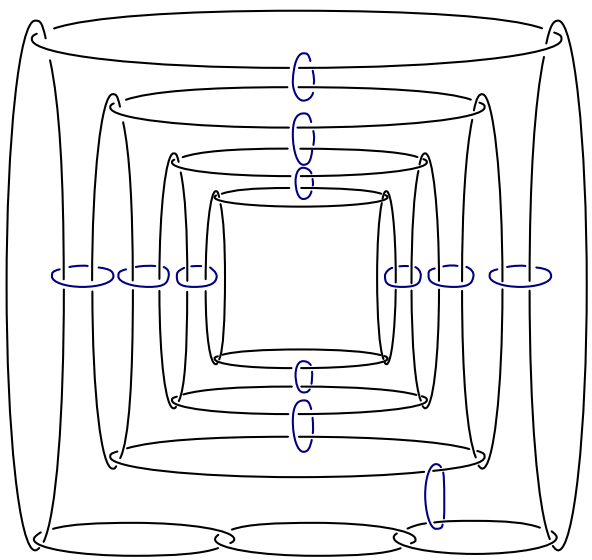


Why does this work?

One direction is immediate: from a graph drawing with low crossing number we get a link diagram with low crossing number.

For the other direction, we want to prove that in any diagram of low crossing number, things are as we would expect:

- the frame is rigid and

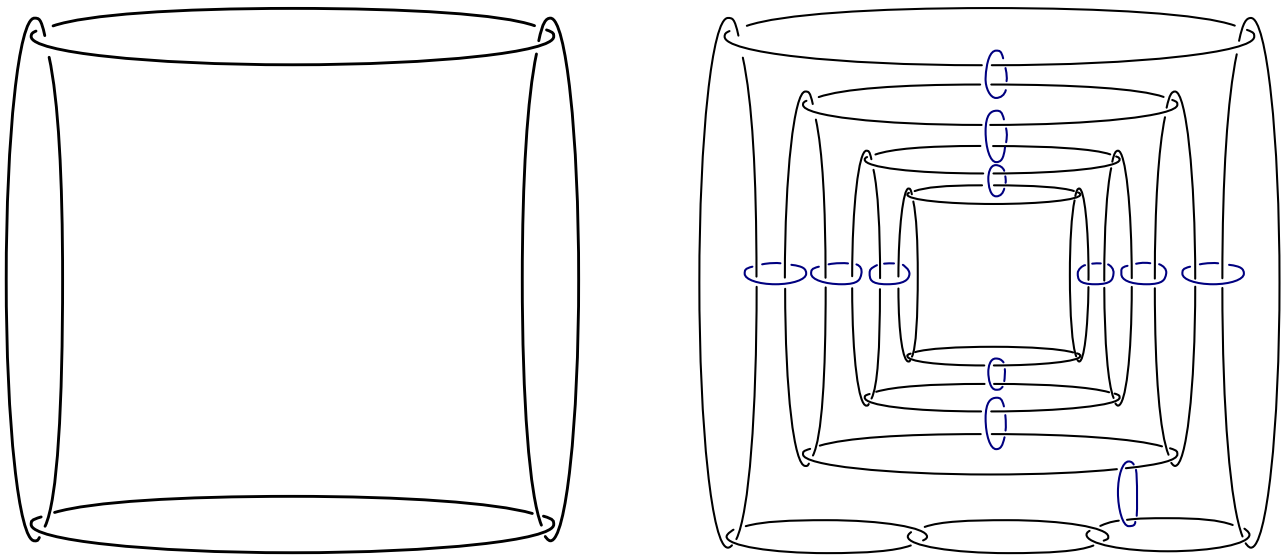


- the only things moving are the red curves.

Using linking numbers

Main tool: Linking numbers.

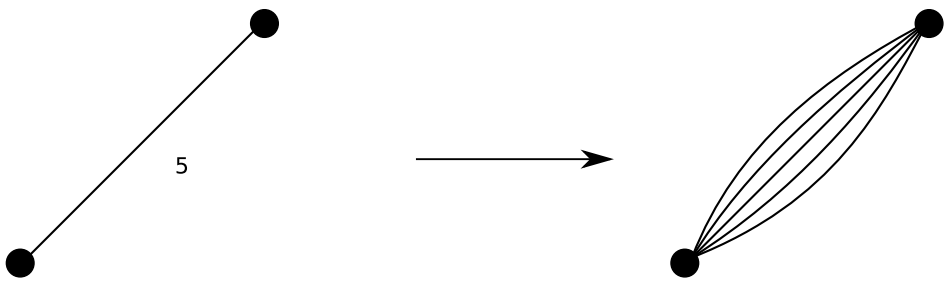
- With linking numbers, we can prove that this diagram of the frame is the unique one with a minimal number of crossings.



- Then the hope is that the placement of the frame forces other crossings (even those not forced by linking numbers).
- But adding the other gadgets may break the rigidity of the frame.

Weighted crossings

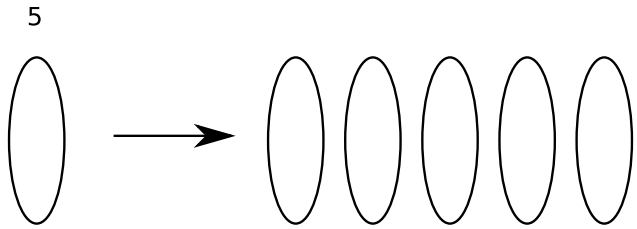
- This is a common issue in reductions involving crossing numbers.
- We can gain rigidity by putting big *weights*: each edges has a weight w_e, and the weighted crossing number of e and f crossing is $w_e w_f$.
- This can be easily simulated by using multiple edges.



- In the setting of graphs, it is immediate that all the multiple edges will be drawn the same way in some crossing-minimal drawing.
- Big weights can enforce rigidity.

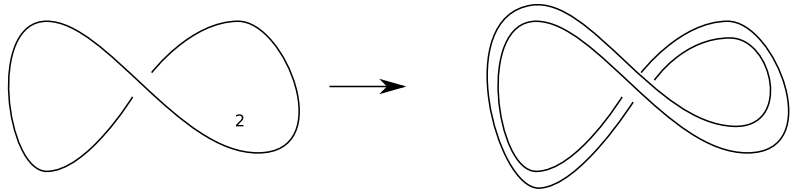
Weighted knots

Likewise, we can use multiple copies of knot to represent weights:

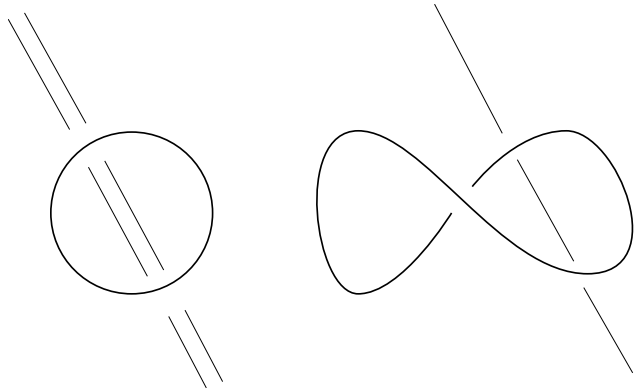


However:

- Self-crossings throw off the accounting, hence we use unknots in the reduction.



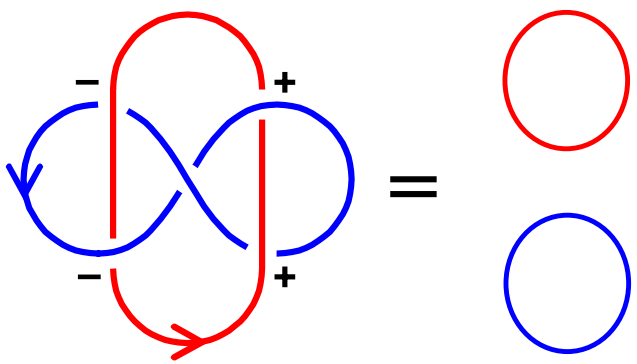
- We can not argue that in a crossing-minimal drawing, all the copies of a knot will be drawn the same way.



Our solution

- We do use weighted knots, and choose weights wisely.
- When arguing that things look like we want them to look, we use a relaxed notion of equivalence.

Two links are *parity-link equivalent* if the parity of the linking number between pairs of components is the same in both crossings.

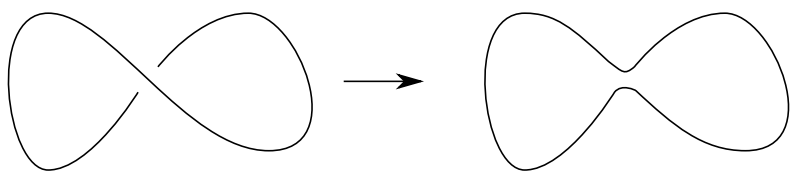


Parity-link equivalence is simpler to handle

Lemma

For any link L, let D' be a diagram with a minimum number of crossings of a link L' which is parity-link equivalent to L. Then no link component in D' has self-crossings.

Proof:



Working from a different link.

- The argument showing that the frame is rigid is only based on linking numbers!

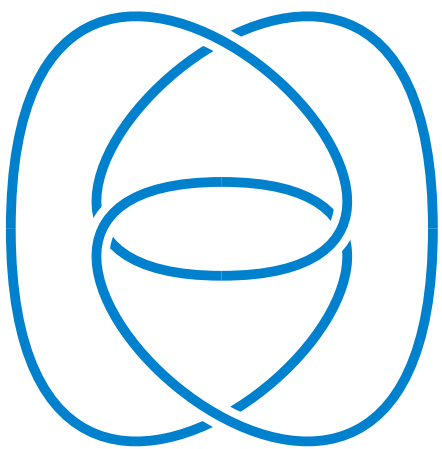
So, if L has a drawing with a low crossing number:

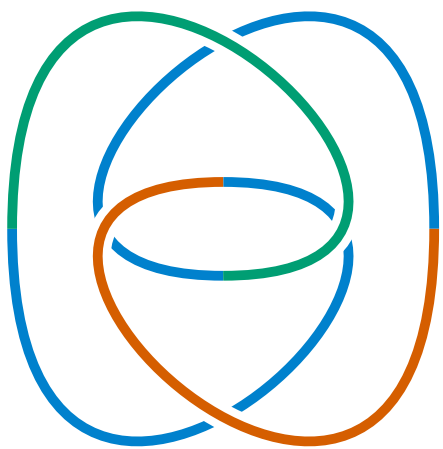
- We look at the crossing-minimal drawing D of a link L' that is parity-link equivalent to our link L. It also has a low number of crossings.
- L' might be different from L, but it does not matter:
- There, the frame is rigid.
- Likewise, the only non-rigid pieces are the moving red curves.
- We can find a drawing of our original bipartite graph from D with few crossings.

We also get **NP**-hardness for the minimal crossing number under other notions of equivalence: parity-link equivalence, linking-number equivalence, link-homotopy and link concordance.

What next?

- How to adapt this to knots? Or links with a bounded number of components? *Alternating knots* might help but the weighting issue is problematic.
- Is it still hard for a fixed value of the crossing number? Note that for a fixed k, determining whether a graph has crossing number at most k can be done in linear-time [Kawarabayashi-Reed'07].
- What about the *bridge number*? (minimum number of *bridges* to draw the knot)





One more speculative slide

What is the computational complexity of knot problems where one forces to stay in the PL category with at most k segments? For example:

- Complexity of the *stick number* (minimum number of segments to realize a knot)?
- Complexity of deciding whether two knots made of k segments can be isotoped to each other using knots made of k segments?

I would expect strong connections to the theory of *linkages* and the *existential theory of the reals*. In particular, is the space of realizations of a knot universal in the sense of Mnev?

Open problem: Stuck geometric unknots, [Calvo '01]

Are there topological unknots which cannot be untangled geometrically in the sense above?

One more speculative slide

What is the computational complexity of knot problems where one forces to stay in the PL category with at most k segments? For example:

- Complexity of the *stick number* (minimum number of segments to realize a knot)?
- Complexity of deciding whether two knots made of k segments can be isotoped to each other using knots made of k segments?

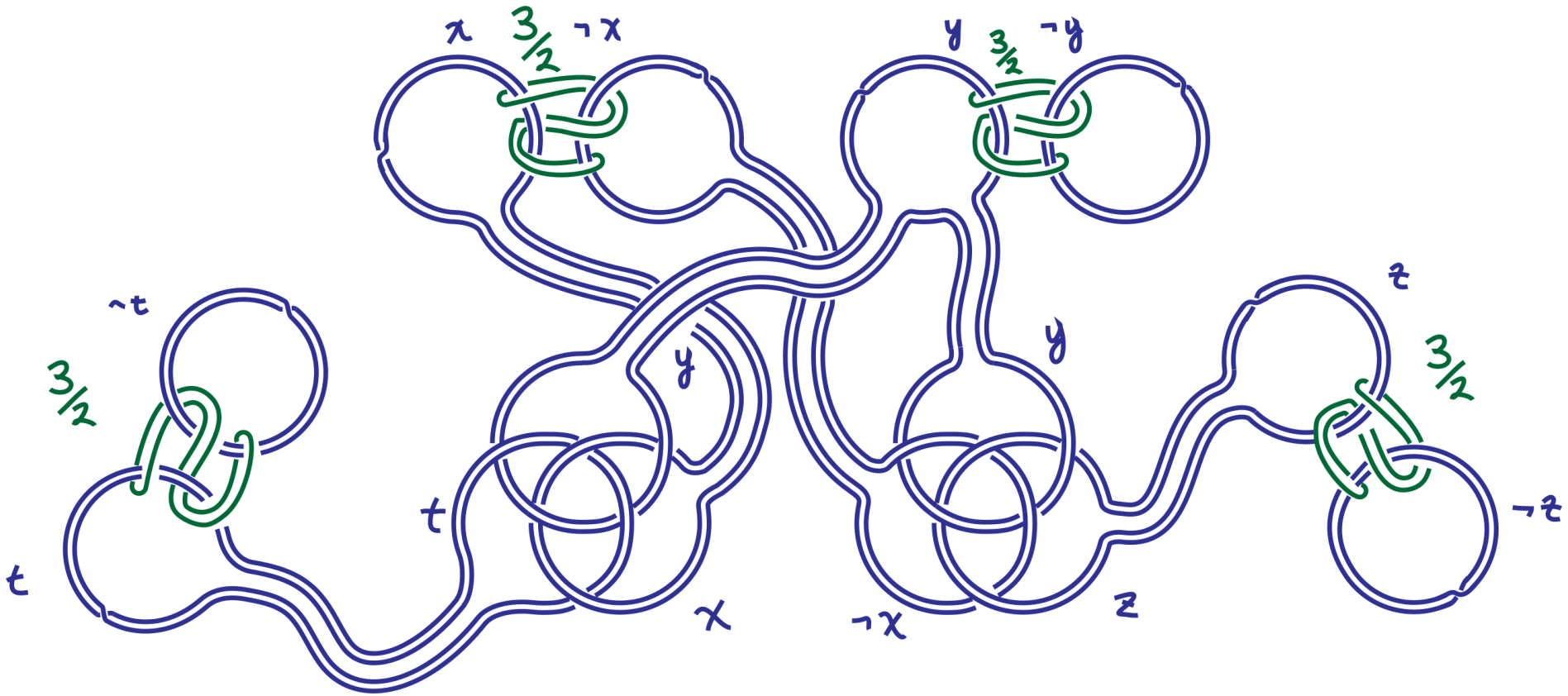
I would expect strong connections to the theory of *linkages* and the *existential theory of the reals*. In particular, is the space of realizations of a knot universal in the sense of Mnev?

Open problem: Stuck geometric unknots, [Calvo '01]

Are there topological unknots which cannot be untangled geometrically in the sense above?

Thank you! Questions?

Our original motivation: $\textsc{Embed}_{2\to 3}$ and $\textsc{Embed}_{3\to 3}$



This 3-manifold embeds into $\mathbb{S}^3$ if and only if Φ is satisfiable.
$\to$ Deciding whether a 3 or a 2-dimensional space embeds into $\mathbb{R}^3$ is **NP**-hard.
Best algorithm runs in a tower of exponentials [Matoušek, Sedgwick, Tancer, Wagner '16].