Knot Diagrams of Treewidth Two

Hans L. Bodlaender[1(✉)], Benjamin Burton[2], Fedor V. Fomin[3], and Alexander Grigoriev[4]

[1] Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
h.l.bodlaender@uu.nl
[2] School of Mathematics and Physics, The University of Queensland, Brisbane, QLD 4072, Australia
[3] Department of Informatics, University of Bergen, 5020 Bergen, Norway
[4] Maastricht University School of Business and Economics, Maastricht, The Netherlands

Abstract. In this paper, we study knot diagrams for which the underlying graph has treewidth two. We give a linear time algorithm for the following problem: given a knot diagram of treewidth two, does it represent the trivial knot? We also show that for a link diagram of treewidth two we can test in linear time if it represents the unlink. From the algorithm, it follows that a diagram of the trivial knot of treewidth 2 can always be reduced to the trivial diagram with at most n untwist and unpoke Reidemeister moves.

Keywords: Knot diagrams · Knot theory · Graph algorithms · Treewidth · Series parallel graphs

1 Introduction

A *knot* is a piecewise linear closed curve S^1 embedded into the 3-sphere S^3 (or the three-dimensional Euclidean space $\mathbb{R}^3$). Two knots are said to be *equivalent* if there is an ambient isotopy between them. In other words, two knots are equivalent if it is possible to distort one knot into the other without breaking it. The basic problem of knot theory is the following unknotting problem: given a knot, determine whether it is equivalent to a knot that bounds an embedded disk in S^3. Such a knot is called the *trivial knot* or simply the *unknot*.

Despite a significant progress, the computational complexity of the unknotting problem remains open. Even the existence of *any* algorithm for this problem

A large part of this research was done during the workshop *Fixed Parameter Computational Geometry* at the Lorentz Center in Leiden. The research of the first author was partially supported by the *Networks* project, supported by the Netherlands Organization for Scientific Research N.W.O. The second author was supported by the Australian Research Council under the Discovery Projects scheme (DP150104108). The third author was supported by the Research Council of Norway via the project "MULTIVAL".

I. Adler and H. Müller (Eds.): WG 2020, LNCS 12301, pp. 80–91, 2020.
https://doi.org/10.1007/978-3-030-60440-0_7

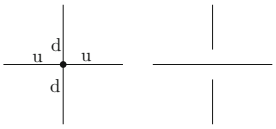

Fig. 1. A vertex of degree four representing a crossing of two strings

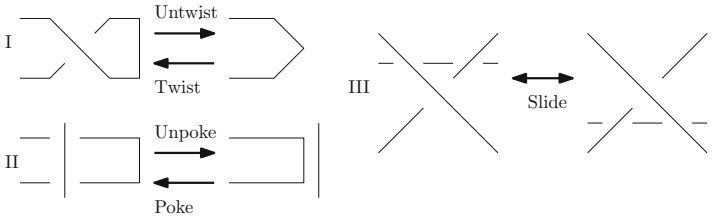

Fig. 2. Reidemeister moves

is a highly non-trivial question. As was stated by Turing in 1954 in [17], "No systematic method is yet known by which one can tell whether two knots are the same." The first algorithm resolving this problem is due to Haken [6]. By the celebrated result of Hass, Lagarias, and Pippenger [8], unknot recognition is in NP. The problem is also in co-NP, see Lackenby [12]. However, no polynomial algorithm for the unknotting problem is known.

It was understood already in 1920s that the question about equivalence of knots in $\mathbb{R}^3$ is reducible to a combinatorial question about knot diagrams [1,14]. Knot diagrams are labeled planar graphs representing a projection of the knot onto a plane. Thus every vertex of the graph in knot diagram is of degree 4 and edges are marked as overcrossing and undercrossing, see Fig. 1 and Sect. 2.

It is one of the most fundamental theorems in knot theory from 1920s that any two diagrams of a knot or link in $\mathbb{R}^3$ differ by a sequence of Reidemeister moves [14], illustrated in Fig. 2. We refer to these moves as (I) twist moves, (II) poke moves, and (III) slide moves, with the reverse operation of a twist move the untwist, and the reverse operation of a poke the unpoke.

With help of Reidemeister moves, see Fig. 2, we obtain an equivalence relation on knot diagrams: if a diagram can be obtained from another by zero or more Reidemeister moves, then these diagrams are equivalent. In our paper, we allow subdivision vertices (i.e., vertices of degree two), and extend the notion of equivalence in the following trivial way: diagrams are equivalent if they can be obtained from another by zero or more Reidemeister moves and additions or removals of subdivisions.

In particular, the diagram of every unknot can be reduced to the trivial diagram (a circle) by performing Reidemeister moves. While each of the Reidemeister moves can be performed in polynomial time, it is very unclear how many of these moves are required to transform an unknot to the trivial diagram. The problem is that sometimes a successful unknotting sequence of Reidemeister moves is not monotone, that is, it has to increase the number of crossings

(vertices) in the knot diagram, see e.g. [9]. Bounding the number of required Reidemeister moves by any function on the number of vertices in the knot diagram was a long-standing open question in the area. The answer to this question was given by Hass and Lagarias [7] who gave the first (exponential) upper bound on the number of Reidemeister moves. Later Lackenby in [11] improved the bound significantly by showing that any diagram of the unknot with n crossings may be reduced to the trivial diagram using at most $(236n)^{11}$ Reidemeister moves. Let us note that this also implies that the unknotting problem is in NP.

In this work we consider the unknotting problem when the given knot diagram has treewidth at most 2. Our main algorithmic result is Theorem 1.

Theorem 1. *Deciding whether any diagram with n crossings and treewidth at most 2 is a diagram of the unknot can be decided in time $O(n)$.*

Our proof yields also the following combinatorial result about the number of Reidemeister moves. It is interesting to note that in Theorem 2 we do not use any slide moves.

Theorem 2. *Any diagram of treewidth 2 of the unknot with n crossings may be reduced to the trivial diagram using at most n untwist and unpoke Reidemeister moves.*

Actually, the techniques developed to prove Theorems 1 and 2 can be used to solve a slightly more general problems about links with diagrams of treewidth 2.

Related Work. To the best of our knowledge, the question whether the unknotting problem with diagrams of bounded treewidth can be resolved in polynomial time is open. Makowsky and Mariño in [13] studied the parametrized complexity of the knot (and link) polynomials known as Jones polynomials, Kauffman polynomials and HOMFLY polynomials on graphs of bounded treewidth. For the Jones and HOMFLY polynomials no example of a non-trivial knot with trivial polynomial is known [4]. Therefore, if e.g. the Jones polynomial recognizes the unknot, then the algorithm from [13] also recognizes the unknot in time FPT in the treewidth.

Rué et al. [16] studied the class of link-types that admit a K_4-minor-free diagram (which is of treewidth at most 2). They obtain counting formulas and asymptotic estimates for the connected K_4-minor-free link and unknot diagrams. While Rué et al. [16] do not discuss algorithms in their work, the combinatorial tools developed in their paper can also be used to obtain Theorem 1. Our approach is more direct, and gives a fairly simple algorithm which is very straightforward to implement. We believe that the notion of double edge is an interesting concept of separate interest. Also, our work was done independently from the work by Rué et al. [16].

Approach. Our main approach is the following. We introduce the notion of *generalized knot diagrams*—these extend knot diagrams with the notion of *double edges* (see Sect. 2). The algorithm starts making the graph simple by adding subdivision vertices. By repeatedly applying *safe reduction rules* (see Sect. 3),

we obtain a series of generalized knot diagrams, that all are equivalent to the input knot diagram, have treewidth at most two, and are simple. This continues till we have classified the (generalized) diagram as knot, unknot, link or unlink. The safe reduction rules come from a well known insight of graphs of treewidth two (Theorem 3), and a case analysis for different types of vertices and edges in the generalized knot diagram. The main algorithm is explained in Sect. 4.

2 Graphs and (Generalized) Knot Diagrams

2.1 Graphs

A *subdivision* in a graph $G = (V, E)$ is a vertex of degree two. The operation to *add a subdivision* is the following: take an edge $\{v, w\}$, and replace this edge by edges $\{v, x\}$ and $\{x, w\}$ with x a new vertex. The operation to *remove* a subdivision is the following: take a vertex of degree 2, add an edge between its neighbors and then remove the vertex and its incident edges.

We do not need to give the definition of treewidth [15], but instead rely on the following well known results on treewidth.

Theorem 3 (Folklore, see e.g., Theorem 33 and Lemma 90 in [2]).

(i) If G has treewidth at most two and is not the empty graph, then G has a vertex of degree at most two.

(ii) If G has treewidth at most two, and G' is obtained from G by removing a vertex, removing an edge, adding a subdivision or removing a subdivision, then the treewidth of G is also at most two.

2.2 Generalized Knot Diagrams

Our algorithm is based upon a generalization of knot diagrams, which we call *generalized knot diagrams*. The main ingredient is a new type of edges, which are created in the course of the algorithm. While a *single* edge in a knot diagram represents a piece of a single string, a *double* edge in a generalized knot diagram represents two pieces of strings between two pairs of vertices of degree two. Specifically, consider any two pieces of strings not intersected by any other piece of string. Let the two pieces of strings have the (four) endpoints at vertices of degree two. Moreover, let the strings alternate at every two consecutive crossings with respect to over- and under-crossing, i.e., if string s is over-crossing string s' at a crossing, then at the next (consecutive) crossing s' is over-crossing s. In accordance with Reidemeister terminology, we refer to these alternating crossings as *twists*. Such two strings with twists between two pairs of vertices of degree two are referred as double edges.

For each double edge we create an integer label that gives the number of twists/crossings in the double edge. If the two pieces of string do not cross, the label is zero. With labelings of endpoints of the strings with u (up) and d (down),

Fig. 3. A three-twist double edge and its string representation

we can distinguish between overcrossings and undercrossings; details are given later in this section.

See Fig. 3 for an illustration how a double edge represents two pieces of string with three twists.

In the generalized knot diagram we identify a pair of degree two vertices associated with an endpoint of a double edge as one *double vertex*, thus creating a new simple graph with a mix of knot diagram (*single*) vertices, double vertices, single and double edges, where double edges are labeled with numbers of twists.

Types of vertices. During the algorithm, we maintain that at each vertex of the diagram, either two or four pieces of string meet. Thus, we have the following types of vertices:

- A vertex of degree one, incident to one double edge (Type 1)
- A vertex of degree two, incident to two double edges (Type 2D)
- A vertex of degree two, incident to two single edges (Type 2S)
- A vertex of degree four, incident to four single edges (Type 4)
- A vertex of degree three, incident to one double edge and two single edges (Type 3)

Type 1, 2D and 3 vertices are called *double vertices*. Each of these is incident to a double edge. It is important to note that we do not have a crossing *at* a double vertex, i.e., all crossings either are at Type 4 vertices or at double edges with a non-zero label.

Each double edge whose integer label is non-zero has u and d-labelings attached to it, that determine the over- and undercrossings for the part of the diagram modelled by this edge. Each endpoint of the edge has a pair consisting of a u and a d attached to it; one of these comes clockwise directly before the edge, and one directly clockwise after the edge. (This can be represented by one bit.)

Thus, the two endpoints at a double vertex of the strings represented by a double edge have labels u and d, respectively. This models that the string labeled by u starts with an overcrossing and the string labeled by d starts with an undercrossing. See Fig. 4 at vertices v, w and x.

In this way, to each generalized knot diagram we can associate a knot diagram: we replace each double edge by a subgraph. If the double edge has integer label i, then we have i vertices of degree four. The u and d labels at these vertices are determined by the d and u labels at the endpoints of the double edge, as explained above. We add where necessary subdivision vertices of degree two to ensure that the graph is simple. We say that two generalized knot diagrams are equivalent when their associated knot diagrams are equivalent.

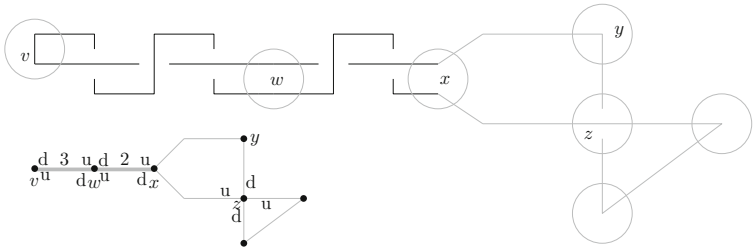

Fig. 4. A knot diagram and a corresponding generalized knot diagram. v is Type 1; w is Type 2D; x is Type 3; y is Type 2S; z is Type 4.

3 Safe Reduction Rules

In this section we introduce a number of reduction rules for generalized knot diagrams. The result of a rule is always again a generalized knot diagram. Note that we always remove one (single or double) vertex of degree at most 2, and possibly add an edge between the neighbors of a removed vertex of degree 2. Thus, when any of these rules is applied, the size of the generalized knot diagram is decreased by at least one vertex.

A rule is *safe*, if whenever we obtain the generalized knot diagram G' from G by applying the rule, we have that:

- The knot diagram associated with G is equivalent to the knot diagram associated with G'. (I.e., it can be obtained from the other by applying Reidemeister moves and adding or removing subdivision vertices.)
- If the treewidth of G is at most two, then the treewidth of G' is at most two.

Note that safeness of a rule implies that application of the rule preserves the ambient isotopy of the original and the resulting knots.

We will show that for all vertices of degree at most two, when G has at least three vertices, we have a safe rule that decreases the number of vertices by at least one, or we can resolve the problem. We have seven cases: vertices of Type 1, 2S, and 2D, where for the latter, their neighbors can be non-adjacent, adjacent by a single edge, or adjacent by a double edge.

Several of the rules have a straightforward case analysis for the markings of double edges. Many of these details can be found in the full version, see [3].

3.1 Vertices of Type 1

The first case is a vertex of Type 1: a vertex incident to one double edge. We can remove this vertex with its incident edge. Safeness follows by observing that the twists on this double edge can be removed with Reidemeister untwists. By removing subdivision vertices, we obtain a knot diagram represented by the diagram obtained by removing v and its incident double edge (Fig. 5).

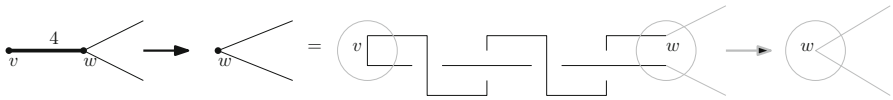

Fig. 5. Removing a Type 1 vertex.

Rule 1. *Let v be of Type 1, incident to double edge $\{v, w\}$. Remove v and its incident edge.*

3.2 Vertices of Type 2S

For a vertex of Type 2S, we have three cases, depending on whether its neighbors are not adjacent, adjacent by a single edge, or adjacent by a double edge. The next Rule 2 is trivially safe as we just remove an edge subdivision. See Fig. 6.

Rule 2. *Let v be incident to two single edges $\{v, w\}$ and $\{v, x\}$, where w and x are not adjacent. Remove v and the edges $\{v, w\}$ and $\{v, x\}$, and add a single edge $\{w, x\}$.*

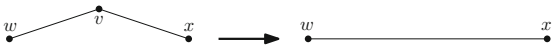

Fig. 6. Removing a vertex with two single edges and non-adjacent neighbors.

The second case is when the neighbors of v are connected by a single edge. This case has a number of different subcases. All are illustrated in Fig. 7.

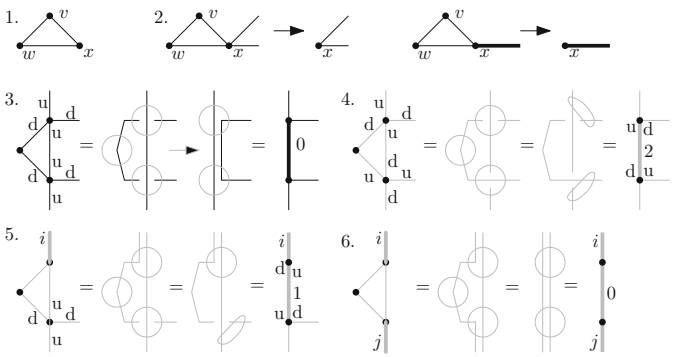

Fig. 7. The cases of Rule 3.

Rule 3. *Let v a Type 2S vertex with neighbors w and x, and suppose $\{w, x\}$ is a single edge.*

1. *If w and x are also Type 2S vertices, then G is a simple cycle of length three, and we do not apply any further rules as the diagram represents the unknot.*
2. *If w is of Type 2S and w of Type 3 or 4, then delete vertices v and w together with their incident edges.*
3. *If w and x are both Type 4 and the $\{u, d\}$-labels of the edge $\{w, x\}$ at vertices w and x are the same, we delete vertex v together with edges $\{v, w\}, \{v, x\}$ and $\{w, x\}$, and we create a double edge $\{w, x\}$ with label 0.*
4. *If w and x are both Type 4 and the the $\{u, d\}$-labels of the edge $\{w, x\}$ at vertices w and x are different, we delete vertex v together with edges $\{v, w\}, \{v, x\}$ and $\{w, x\}$, and we create a double edge $\{w, x\}$ of label 2.*
5. *If w is of Type 4 and x is of Type 3, we delete vertex v together with edges $\{v, w\}, \{v, x\}$ and $\{w, x\}$, and create a double edge $\{w.x\}$ with label 1.*
6. *If both x and w are of Type 3, we delete vertex v together with edges $\{v, w\}$, $\{v, x\}$ and $\{w, x\}$, and we create a double edge $\{w, x\}$ with label 0.*

Equivalence of the diagrams before and after the reduction step is easy to see. In Case 2, when w is of Type 4, we do one untwist removing the crossing at w; in Case 3, we do one unpoke. In the other cases, we do not perform Reidemeister moves, but by removing subdivisions, we can replace the generalized knot diagram by one with fewer vertices that represents the same knot diagram.

We now look at the third case. Suppose v is adjacent by two single edges to two double vertices, w and x, and there is a double edge between w and x with label i, i.e., having i twists.

Rule 4. *Suppose v is of Type 2S with neighbors w and x, and there is a double edge between w and x with i twists.*

1. *If $i = 0$, then the generalized knot diagram represents an unlink. We recurse on the generalized diagram obtained by removing v and incident edges, and making the edge $\{w, x\}$ a single edge.*
2. *If $i \neq 1$ is odd, the generalized knot diagram represents a non-trivial knot;*
3. *If $i \neq 0$ is even, the generalized knot diagram represents a non-trivial link.*
4. *If $i = 1$, then delete vertex v together with adjacent edges and delete double edge $\{w, x\}$, make w and x single vertices adjacent by a single edge.*

The cases are illustrated in Fig. 8. Correctness of the first three cases is evident. Safeness of the fourth case follows as this step represents a single Reidemeister untwist with subsequent contraction of subdivision.

3.3 Vertices of Type 2D

For vertices of Type 2D (incident to two double edges), we have again three cases: the neighbors are not adjacent, the neighbors are adjacent by a single edge, or the neighbors are adjacent by a double edge.

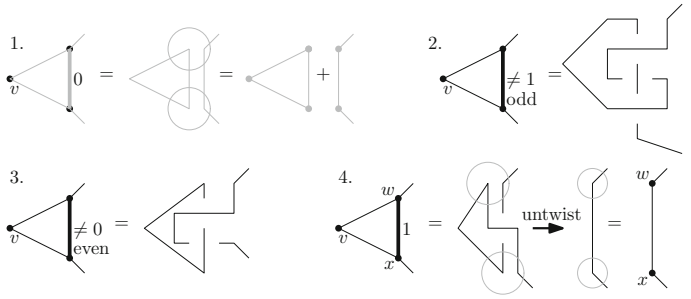

Fig. 8. The cases of Rule 4.

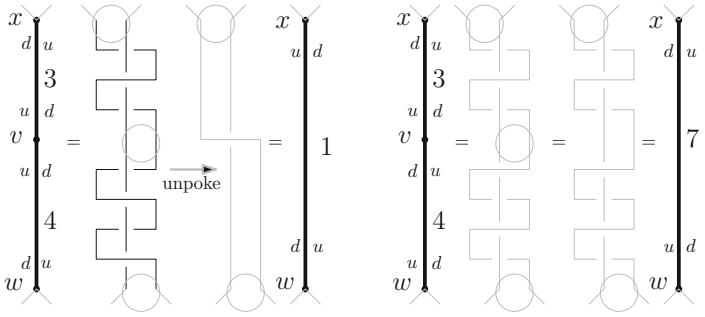

Fig. 9. Rule 5 with agreeing (left) and disagreeing (right) labels at v.

Rule 5. *Let v be of Type 2D where neighbors w and x are not adjacent. Suppose $\{v, w\}$ has label i, and $\{v, x\}$ has label j. Remove v and the edges $\{v, w\}$ and $\{v, x\}$, and add a double edge $\{w, x\}$.*

1. *If $i = 0$ ($j = 0$), the double edge $\{w, x\}$ gets label j (i), and the $\{u, d\}$-labels are as for the edge $\{v, x\}$ ($\{v, w\}$).*
2. *If the $\{u, d\}$-labels at v are at both sides equal (the left case in Fig. 9), then the double edge $\{w, x\}$ gets label $|i - j|$. If $i \neq j$, keep the $\{u, d\}$-labels at the side of the larger number i or j, and switch them at the other side.*
3. *If the $\{u, d\}$-labels at v differ at each of the sides (the right case in Fig. 9), then the double edge gets label $i + j$. Set the labels of the new double edge at w and x in the same way as the original double edges of these vertices to v.*

Lemma 1. *Rule 5 is safe.*

Proof. In the case of agreement on labels, see Fig. 9 (left), we proceed with $\min\{i, j\}$ Reidemeister unpoke moves followed by removing the subdivision. In case of label disagreement, see Fig. 9 (right), we keep exactly the same knot diagram, but simplify the generalized knot diagram by removing the subdivision on the double edge. In the latter case the number of twists on the new double edge is exactly the sum $i + j$ of the numbers of twists on the two original double edges. □

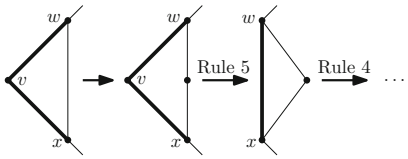

Fig. 10. Rule 6 is reduced to rule Rule 4

Rule 6. *Let v be of Type 2D, where neighbors w and x are adjacent by a single edge. This case can be handled with help of earlier cases. First, we add a subdivision of $\{w, x\}$. Then, we apply Rule 5 arriving in and applying one of the cases of Rule 4, see Fig. 10. Therefore, we either classify the diagram or safely reduce the graph.*

Rule 7. *Consider three double vertices v, w and x with three double edges, $\{v, w\}$, $\{x, v\}$ and $\{w, x\}$ of labels i, j and k, respectively. Apply Rule 5 to w, but instead of removing w, we set the new double edge (with label $i + k$ or $|i - k|$, say ℓ) to be $\{v, w\}$, and let $\{w, x\}$ be a double edge with label 0. Now, apply Rule 5 to v, but instead of removing v, we set the new double edge (with label $j + \ell$ or $|j - \ell|$, say m) to be $\{v, w\}$, and let $\{v, x\}$ be a double edge with label 0. Depending on the value of m, we can classify the knot diagram.*

1. *If $m = 0$, then the generalized knot diagram represents the unlink.*
2. *If $m \neq 1$ is odd, the generalized knot diagram represents the $(m, 2)$-torus knot, for definition and notations see [18].*
3. *If $m \neq 0$ is even, the generalized knot diagram represents the $(m, 2)$-torus link.*
4. *If $m = 1$, then the generalized knot diagram represents the unknot as a single Reidemeister untwist turns the diagram to a circle (see Fig. 11, right.)*

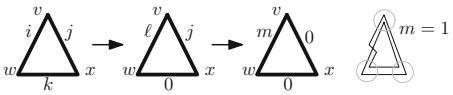

Fig. 11. Illustration to Rule 7

4 Main Algorithm

The main algorithm starts by subdividing where necessary edges to obtain a simple graph. Then, while there are at least three vertices, we repeatedly take a vertex that is incident to at most two edges, and apply a safe rule. Each rule application decreases the number of vertices, so after $O(n)$ such steps we

resolve the problem. Standard techniques (for details see [3]) give a linear time implementation. Note that the safe rules only execute untwists and unpokes, but no other Reidemeister moves. Each untwist and unpoke decreases the number of twists by at least one. Thus we have shown Theorems 1 and 2. Next to these main results we have the following two straightforward corollaries.

Corollary 1. *Any knot/link having a diagram of treewidth 2 is a knot sum of* $(\cdot, 2)$*-torus knots/links.*

Corollary 2 (See [3]). *Given two knot diagrams of treewidth 2, the equivalence of the knots is verifiable in linear time (by simply comparing the resulting torus knots in Rules 4 and 7).*

5 Conclusions

We conclude with the following questions.

- We gave a linear time algorithm deciding whether any diagram with n crossings and treewidth 2 is a diagram of the unknot. The question arises: Is it possible to extend our result to graphs of treewidth $t \geq 3$? Even the existence of a polynomial time algorithm for $t = 3$ is open. Extension of our results to the graphs of treewidth $t = 3$ requires new arguments and techniques: Our algorithm monotonically decreases the number of crossings in a treewidth 2 diagram as only untwist and unpoke Reidemeister moves are performed, while there are unknot diagrams of treewidth 3 requiring increase of the number of crossings for unknotting, e.g., the *Culprit*, the *Goeritz unknot* and some other small but hard unknots [9].
- Koenig and Tsvietkova [10] conjectured and de Mesmay et al. [5] proved that deciding if a diagram of the unknot can be untangled using at most k Reidemeister moves (where k is part of the input) is NP-hard. Could this problem be solved in polynomial time on knots with diagrams of treewidth 2?

References

1. Alexander, J.W., Briggs, G.B.: On types of knotted curves. Ann. Math. (2) **28**(1–4), 562–586 (1926). https://doi.org/10.2307/1968399
2. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theoret. Comput. Sci. **209**, 1–45 (1998)
3. Bodlaender, H.L., Burton, B.A., Fomin, F.V., Grigoriev, A.: Knot diagrams of treewidth two (2019). http://arxiv.org/abs/1904.03117
4. Dasbach, O.T., Hougardy, S.: Does the Jones polynomial detect unknottedness? Exp. Math. **6**(1), 51–56 (1997)
5. de Mesmay, A., Rieck, Y.A., Sedgwick, E., Tancer, M.: The unbearable hardness of unknotting. In: 35th International Symposium on Computational Geometry, SoCG 2019, LIPIcs, vol. 129, pp. 49:1–49:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)

6. Haken, W.: Theorie der Normalflächen. Acta Math. **105**, 245–375 (1961). https://doi.org/10.1007/BF02559591
7. Hass, J., Lagarias, J.C.: The number of Reidemeister moves needed for unknotting. J. Am. Math. Soc. **14**(2), 399–428 (2001). https://doi.org/10.1090/S0894-0347-01-00358-7
8. Hass, J., Lagarias, J.C., Pippenger, N.: The computational complexity of knot and link problems. J. ACM **46**(2), 185–211 (1999). https://doi.org/10.1145/301970.301971
9. Henrich, A., Kauffman, L.H.: Unknotting unknots. arXiv preprint arXiv:1006.4176 (2010)
10. Koenig, D., Tsvietkova, A.: NP-hard problems naturally arising in knot theory. arXiv preprint arXiv:1809.10334 (2018)
11. Lackenby, M.: A polynomial upper bound on Reidemeister moves. Ann. Math. **182**(2), 491–564 (2015). https://doi.org/10.4007/annals.2015.182.2.3
12. Lackenby, M.: The efficient certification of knottedness and thurston norm. arXiv preprint arXiv:1604.00290 (2016)
13. Makowsky, J.A., Mariño, J.P.: The parameterized complexity of knot polynomials. J. Comput. Syst. Sci. **67**, 742–756 (2003)
14. Reidemeister, K.: Knoten und Gruppen. Abh. Math. Sem. Univ. Hamburg **5**(1), 7–23 (1927). https://doi.org/10.1007/BF02952506
15. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. J. Algorithms **7**, 309–322 (1986)
16. Rué, J., Thilikos, D.M., Velona, V.: Structure and enumeration of K_4-minor-free links and link diagrams. Electron. Notes Discret. Math. **68**, 119–124 (2018). https://doi.org/10.1016/j.endm.2018.06.021
17. Turing, A.M.: Solvable and Unsolvable Problems. Penguin Books, London (1954)
18. Torus knot (2019). http://mathworld.wolfram.com/TorusKnot.html