Computing Optimal Tangles Faster

Oksana Firman*[1], Philipp Kindermann[1], Alexander Ravsky†[2], Alexander Wolff‡[1], and Johannes Zink[1]

1 **Institut für Informatik, Universität Würzburg**
`firstname.lastname@uni-wuerzburg.de`

2 **Pidstryhach Institute for Applied Problems of Mechanics and Mathematics, National Academy of Sciences of Ukraine, Lviv, Ukraine**
`alexander.ravsky@uni-wuerzburg.de`

Abstract

We study the following combinatorial problem. Given a set of n y-monotone *wires*, a *tangle* determines the order of the wires on a number of horizontal *layers* such that the orders of the wires on any two consecutive layers differ only in swaps of neighboring wires. Given a multiset L of *swaps* (that is, unordered pairs of numbers between 1 and n) and an initial order of the wires, a tangle *realizes* L if each pair of wires changes its order exactly as many times as specified by L. The aim is to find a tangle that realizes L using the smallest number of layers. We show that this problem is NP-hard, and we give an algorithm that computes an optimal tangle for n wires and a given list L of swaps in $O((2|L|/n^2+1)^{n^2/2}\varphi^n n)$ time, where $\varphi \approx 1.618$ is the golden ratio. We can treat lists where every swap occurs at most once in $O(n!\varphi^n)$ time. We implemented the algorithm for the general case and compared it to an existing algorithm.

1 Introduction

Our research is based on a recent paper of Olszewski et al. [4] who use *tangles* (which they call *templates*) to visualize chaotic attractors, which occur in chaotic dynamic systems. Such systems are considered in physics, celestial mechanics, electronics, fractals theory, chemistry, biology, genetics, and population dynamics. In the framework of Olszewski et al., one is given a set of wires that hang off a horizontal line in a fixed order, and a multiset of swaps between the wires; a tangle then is a visualization of these swaps, i.e., an order in which the swaps are performed, where only adjacent wires can be swapped and disjoint swaps can be done simultaneously. Olszewski et al. gave an algorithm for minimizing the height of a tangle. They didn't analyze the asymptotic running time of their algorithm (which we estimate below), but tested it on a set of benchmarks.

Wang [5] used the same optimization criterion for tangles, given only the final permutation. She showed that, in an optimal tangle, no swap occurs more than once. She used odd-even sort, a parallel variant of bubble sort, to compute tangles with at most one layer more than the minimum. Bereg et al. [1, 2] showed, given a final permutation, how to minimize the number of bends or *moves* (which are maximal "diagonal" segments of the wires).

Framework, Terminology, and Notation. We modify the terminology of Olszewski et al. [4] in order to introduce a formal algebraic framework for the problem. Given a natural number n of wires, a *(swap) list* $L = (l_{ij})$ of *order* n is a symmetric $n \times n$ matrix with non-negative

* Supported by DAAD.
† Supported by Erasmus+.
‡ Supported by DFG grant WO 758/9-1.

$$L_n = \begin{pmatrix} 0 & 1 & 1 & \dots & 1 & 0 & 2 \\ 1 & 0 & 1 & \dots & 1 & 2 & 0 \\ 1 & 1 & 0 & \dots & 1 & 0 & 2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & 0 & \mathbf{0} & \mathbf{2} \\ 0 & 2 & 0 & \dots & \mathbf{0} & 0 & n-1 \\ 2 & 0 & 2 & \dots & \mathbf{2} & n-1 & 0 \end{pmatrix}$$

(The bold zeros and twos must be swapped if n is even.)

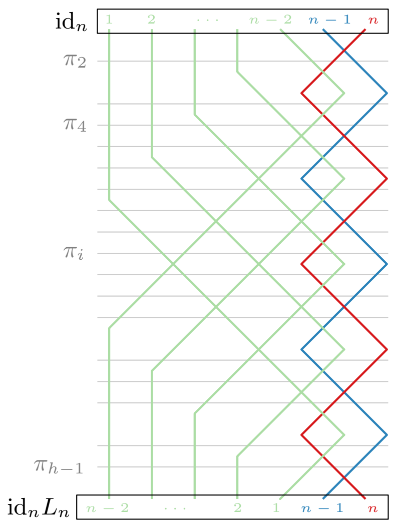

Figure 1 A list L_n for n wires (left) and the unique tangle of minimum height realizing L_n (right) for the start permutation $\mathrm{id}_n = 123\dots n$. Here, $n = 7$. The tangle is not simple because $\pi_2 = \pi_4$.

entries and zero diagonal. The *length* of L is defined as $|L| = \sum_{i<j} l_{ij}$. A list $L' = (l'_{ij})$ is a *sublist* of L if $l'_{ij} \le l_{ij}$ for each $i, j \in [n]$. A list is *simple* if all its entries are zeroes or ones.

A *permutation* is a bijection of the set $[n] = \{1, \dots, n\}$ onto itself. The set S_n of all permutations of the set $[n]$ is a group whose multiplication is a composition of maps (i.e., $(\pi\sigma)(i) = \pi(\sigma(i))$ for each pair of permutations $\pi, \sigma \in S_n$ and each $i \in [n]$). The identity of the group S_n is the identity permutation id_n. We write a permutation $\pi \in S_n$ as the sequence of numbers $\pi^{-1}(1)\pi^{-1}(2)\dots\pi^{-1}(n)$. For instance, the permutation π of [4] with $\pi(1) = 3$, $\pi(2) = 4$, $\pi(3) = 2$, and $\pi(4) = 1$ is written as 4312. We denote the set of all permutations of order 2 in S_n by $S_{n,2}$, i.e., $\pi \in S_{n,2}$ if and only if $\pi\pi = \mathrm{id}_n$, e.g., $2143 \in S_{4,2}$.

For $i, j \in [n]$, the *swap* ij is the permutation that exchanges i and j, whereas the other elements of $[n]$ remain fixed. A set S of swaps is *disjoint* if each element of $[n]$ participates in at most one swap of S. Therefore, the product $\prod S$ of all elements of a disjoint set S of swaps does not depend on the order of factors and belongs to $S_{n,2}$. Conversely, for each permutation $\varepsilon \in S_{n,2}$ there exists a unique disjoint set $S(\varepsilon)$ of swaps such that $\varepsilon = \prod S(\varepsilon)$.

A permutation $\pi \in S_n$ *supports* a permutation $\varepsilon \in S_{n,2}$ if, for each swap $ij \in S(\varepsilon)$, i and j are neighbors in π. By induction, we can easily show that any permutation $\pi \in S_n$ supports exactly $F_{n+1} - 1$ permutations of order 2 where F_n is the n-th Fibonacci number.

Permutations π and σ are *adjacent* if there exists a permutation $\varepsilon \in S_{n,2}$ such that π supports ε and $\sigma = \pi\varepsilon$. In this case, $\sigma\varepsilon = \pi\varepsilon\varepsilon = \pi$ and σ supports ε, too. A *tangle* T of *height* h is a sequence $\langle \pi_1, \pi_2, \dots, \pi_h \rangle$ of permutations in which every two consecutive permutations are adjacent. A *subtangle* of T is a sequence $\langle \pi_k, \pi_{k+1}, \dots, \pi_l \rangle$ of consecutive permutations of T. Let $L(T) = (l_{ij})$ be the symmetric $n \times n$ matrix with zero diagonal where l_{ij} is the number of occurrences of swap ij in T. We say that T *realizes* $L(T)$; see Fig. 1. A list is *π-feasible* if it can be realized by a tangle starting from a permutation π. An id_n-feasible list is *feasible*; e.g., the list defined by the swaps 13 and 24 is *not* feasible.

A list $L = (l_{ij})$ also can be considered as a multiset of swaps, where l_{ij} is the multiplicity of swap ij. In particular, the notation $ij \in L$ means $l_{ij} > 0$. A tangle is *simple* if all its permutations are distinct. In particular, the height of a simple tangle is at most $n!$.

For each permutation $\pi \in S_n$ and a list $L = (l_{ij})$, we define a map $\pi L\colon [n] \to \mathbb{Z}$,

$$i \mapsto \pi(i) + |\{j\colon \pi(i) < \pi(j) \leq n \text{ and } l_{ij} \text{ is odd}\}| - |\{j\colon 1 \leq \pi(j) < \pi(i) \text{ and } l_{ij} \text{ is odd}\}|.$$

For each wire $i \in [n]$, $\pi L(i)$ is the position of the wire after all swaps in L have been applied to π. A list L is called *π-consistent* if $\pi L \in S_n$, or, more rigorously, if πL induces a permutation of $[n]$. An id_n-consistent list is *consistent*. For example, the list $\{12, 23, 13\}$ is consistent, whereas the list $\{13\}$ isn't. If L is not consistent, then it is clearly not feasible. For a list $L = (l_{ij})$, we define $1(L) = (l_{ij} \bmod 2)$. Since $\mathrm{id}_n L = \mathrm{id}_n 1(L)$, the list L is consistent if and only if $1(L)$ is consistent. We can compute $1(L)$ and check its consistency in $O(n + |1(L)|) = O(n^2)$ time. Hence, in the sequel we assume that all lists are consistent.

The *height* $h(L)$ of a feasible list L is the minimum height of a tangle that realizes L. A tangle T is *optimal* if $h(T) = h(L(T))$. In the Tangle-Height Minimization problem, we are given a swap list L and the goal is to compute an optimal tangle T realizing L. As initial wire order, we always assume the identity id_n.

Our Contribution. We show that Tangle-Height Minimization is NP-hard (see Section 2). We give an exact algorithm for simple lists running in $O(n!\varphi^n)$ time and an exact algorithm for general lists running in $O((2|L|/n^2 + 1)^{n^2/2}\varphi^n n)$ time, which is polynomial in $|L|$ for any fixed $n \geq 2$ (see Section 3). We implemented the algorithm for general lists and compared it to the algorithm of Olszewski et al. [4] using their benchmark set (see Section 4).

In order to be able to also compare the asymptotic runtime behaviors, we now analyze the algorithm of Olszewski et al. [4]. Their algorithm constructs a search tree whose height is bounded by the height $h(L)$ of an optimal tangle for the given list L. The tree has $1 + d + d^2 + \dots + d^{h(L)-1} = (d^{h(L)} - 1)/(d-1)$ vertices, where $d = F_{n+1} - 1$ is a bound on the number of edges leaving a vertex, $F_n = (\varphi^n - (-\varphi)^{-n})/\sqrt{5} \in O(\varphi^n)$ is the n-th number in the Fibonacci sequence, and $\varphi = \frac{\sqrt{5}+1}{2} \approx 1.618$ is the golden ratio. Since it takes $O(n)$ time to deal with each vertex, the total running time is $O(\varphi^{(n+1)(h(L)-1)}5^{-(h(L)-1)/2}n)$. Since $2|L|/n \leq h(L) - 1 \leq |L|$, this time is not better than $O(\varphi^{2|L|}5^{-|L|/n}n)$, which is exponential with respect to $|L|$ for fixed $n \geq 2$ and, hence, slower than our algorithm for the general case.

It is known (see, for instance, Wang [5]) that, for any simple list L, $h(L) \leq n + 1$. This implies that, on simple lists, the algorithm of Olszewski et al. runs in $O(\varphi^{(n+1)n}5^{-n}n) = e^{O(n^2)}$ time, whereas our algorithm for simple lists runs in $O(n!\varphi^n) = e^{O(n \log n)}$ time.

2 Complexity

▶ **Theorem 1.** *Given a list L of swaps and an integer $h > 0$, it is NP-hard to decide whether there is a tangle T of height $h(T) \leq h$ realizing L.*

Proof sketch; for the full proof see [3]. From the given 3-Partition instance A, we construct in polynomial time a list L that can be realized by a tangle of height at most $mMB + O(m^2)$ if and only if A is a yes-instance. For an example instance, see Fig. 2.

In the list L we use two central wires ω and ω' swapping $2m$ times. Two consecutive swaps form a *loop*. We number the loops from top to bottom; depending on their index, loops are *even* or *odd*. We use wires $\beta_i, \beta'_i, \gamma_i, \gamma'_i, \delta_i, \delta'_i$ with $i \in [m]$ to enforce the following. For any tangle T realizing L, the height of the subtangle from the beginning of T to the end of the i-th odd loop is at least $(i-1)MB$ and the height of the subtangle from the $(i+1)$-th odd loop to the end of T is at least $(m-i)MB$, where $M \in \Theta(m^3)$ is some large scaling factor. For a tangle T to not exceed the maximum height, every even loop in T must have a

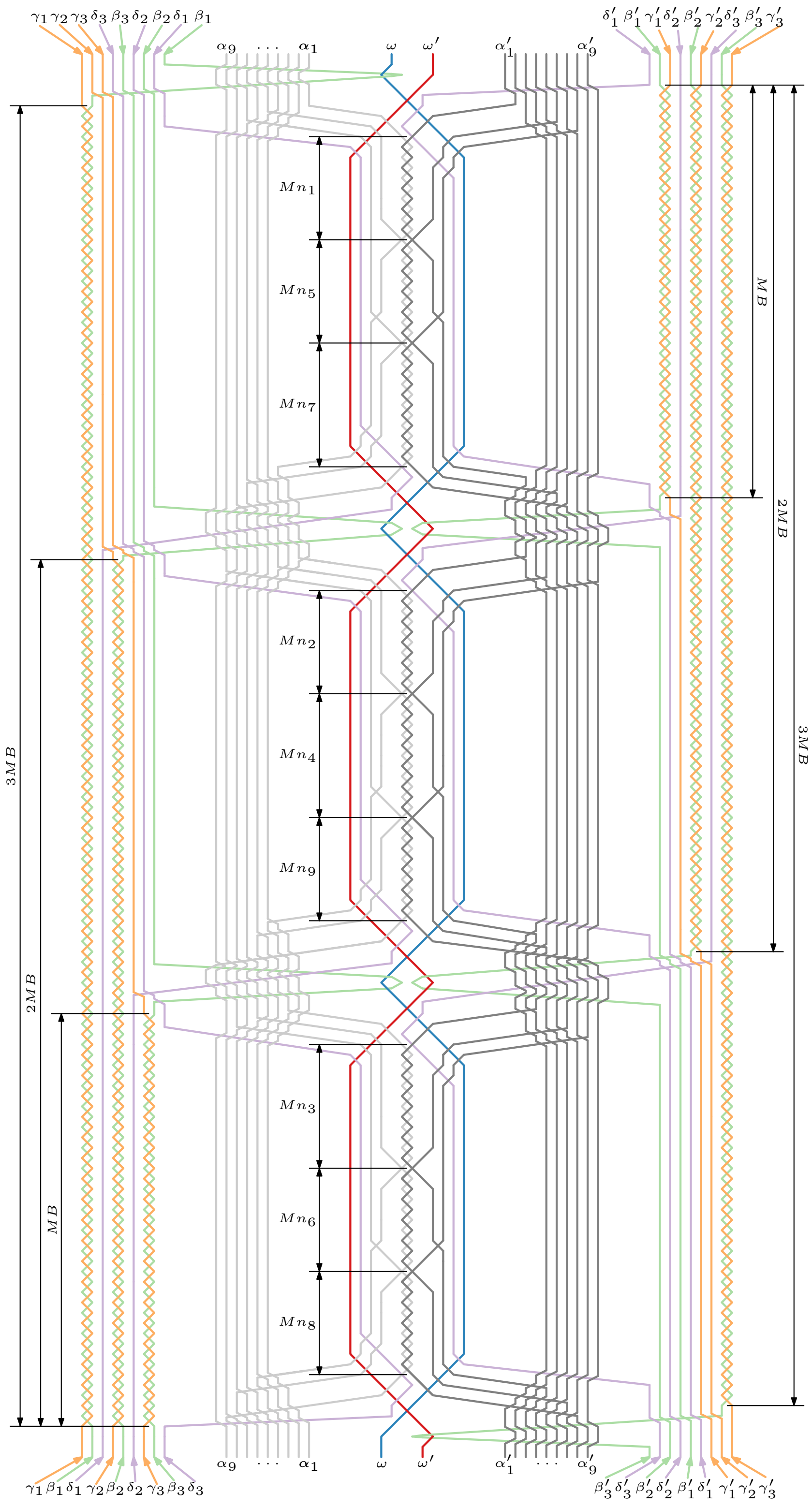
$\gamma_1\gamma_2\gamma_3\delta_3\beta_3\delta_2\beta_2\delta_1\beta_1$
α_9 . . . α_1
ω ω'
α'_1 α'_9
$\delta'_1\beta'_1\gamma'_1\delta'_2\beta'_2\gamma'_2\delta'_3\beta'_3\gamma'_3$
Mn_1
Mn_5
Mn_7
Mn_2
Mn_4
Mn_9
Mn_3
Mn_6
Mn_8
$3MB$
$2MB$
MB
MB
$2MB$
$3MB$
α_9 . . . α_1
ω ω'
α'_1 . . . α'_9
$\gamma_1\beta_1\delta_1\gamma_2\beta_2\delta_2\gamma_3\beta_3\delta_3$
$\beta'_3\delta'_3\beta'_2\delta'_2\beta'_1\delta'_1\gamma'_1\gamma'_2\gamma'_3$

Figure 2 Example of our reduction from 3-Partition to Tangle-Height Minimization with $A_1 = \{n_1, n_5, n_7\}$, $A_2 = \{n_2, n_4, n_9\}$, $A_3 = \{n_3, n_6, n_8\}$, $m = 3$, $B = \sum_{i=1}^{3m} n_i/m$, and $M = 2m^3$.

height of about MB. We encode the numbers in A by introducing, for each $i \in [3m]$, two wires α_i and α'_i that swap Mn_i times. All α_i–α'_i swaps must occur inside exactly one of the even loops, but on different layers. The combination of these blocks of swaps inside the even loops corresponds to a partition of the given 3-Partition instance A. All tangles of height at most $mMB + O(m^2)$ correspond to a solution of A, and if there is a solution of A then there also is a tangle of height at most $mMB + O(m^2)$ realizing this solution. ◀

3 Exact Algorithms

The two algorithms that we describe in this section test whether a given list is feasible and, if yes, construct an optimal tangle realizing the list. For any permutation $\pi \in S_n$, we define the simple list $L(\pi) = (l_{ij})$ such that, for $0 \le i < j \le n$, $l_{ij} = 0$ if $\pi(i) < \pi(j)$, and $l_{ij} = 1$ otherwise. We use the following two lemmas, which we prove in the full version [3].

▶ **Lemma 2.** *For every permutation $\pi \in S_n$, $L(\pi)$ is the unique simple list with $\mathrm{id}_n\, L(\pi) = \pi$.*

▶ **Lemma 3.** *For every tangle $T = \langle \pi_1, \pi_2, \dots, \pi_h \rangle$, we have $\pi_1 L(T) = \pi_h$.*

Simple lists. Let L be a consistent simple list. Wang's algorithm [5] creates a simple tangle from $\mathrm{id}_n\, L$, so L is feasible. Let $T = (\mathrm{id}_n = \pi_1, \pi_2, \dots, \pi_h = \mathrm{id}_n\, L)$ be any tangle such that $L(T)$ is simple. Then, by Lemma 3, $\mathrm{id}_n\, L(T) = \pi_h$. By Lemma 2, $L(\pi_h)$ is the unique simple list with $\mathrm{id}_n\, L(\pi_h) = \pi_h = \mathrm{id}_n\, L$, so $L(T) = L(\pi_h) = L$ and thus T is a realization of L.

We compute an optimal tangle realizing $L = (l_{ij})$ as follows. Consider the graph G_L whose vertex set $V(G_L)$ consists of all permutations $\pi \in S_n$ with $L(\pi) \le L$ (componentwise). A directed edge (π, σ) between vertices $\pi, \sigma \in V(G_L)$ exists if and only if π and σ are adjacent as permutations and $L(\pi) \cap L(\pi^{-1}\sigma) = \varnothing$; the latter means that the set of (disjoint) swaps whose product transforms π to σ cannot contain swaps from the set whose product transforms id_n to π. The graph G_L has at most $n!$ vertices and maximum degree $F_{n+1} - 1 = O(\varphi^n)$, see Section 1. Furthermore, for each $h \ge 0$, there is a natural bijection between tangles of height $h+1$ realizing L and paths of length h in the graph G_L from the initial permutation id_n to $\mathrm{id}_n\, L$. A shortest such path can be found by BFS in $O(E(G_L)) = O(n!\varphi^n)$ time.

▶ **Theorem 4.** *For a simple list of order n,* Tangle-Height Minimization *can be solved in $O(n!\varphi^n)$ time.*

General lists. We can assume that $|L| \ge n/2$; otherwise, there is a wire $k \in [n]$ that doesn't belong to any swap. This wire splits L into smaller lists with independent realizations. (If there is a swap ij with $i < k < j$, then L is infeasible.)

Let $L = (l_{ij})$ be the given list. We compute an optimal tangle realizing L (if it exists) as follows. Let λ be the number of distinct sublists of L. We consider them in order of increasing length. Let L' be the next list to consider. We first check its consistency by computing the map $\mathrm{id}_n\, L'$. If L' is consistent, then we compute an optimal realization $T(L')$ of L' (if it exists), adding a permutation $\mathrm{id}_n\, L'$ to the end of a shortest tangle $T(L'') = \langle \pi_1, \dots, \pi_h \rangle$ with π_h adjacent to $\mathrm{id}_n\, L'$ and $L'' + L(\langle \pi_h, \mathrm{id}_n\, L' \rangle) = L'$. This search also checks the feasibility of L' because such a tangle $T(L')$ exists if and only if the list L' is feasible. Since there are $F_{n+1} - 1$ permutations adjacent to $\mathrm{id}_n\, L'$, we have to check at most $F_{n+1} - 1$ lists L''. Hence, in total we spend $O(\lambda(F_{n+1} - 1)n)$ time for L. Assuming that $n \ge 2$, we bound λ as follows.

$$\lambda = \prod_{i<j} (l_{ij} + 1) \le \left(\frac{\sum_{i<j} (l_{ij} + 1)}{\binom{n}{2}} \right)^{\binom{n}{2}} = \left(\frac{|L|}{\binom{n}{2}} + 1 \right)^{\binom{n}{2}} \le \left(\frac{2|L|}{n^2} + 1 \right)^{n^2/2} \le e^{|L|}.$$

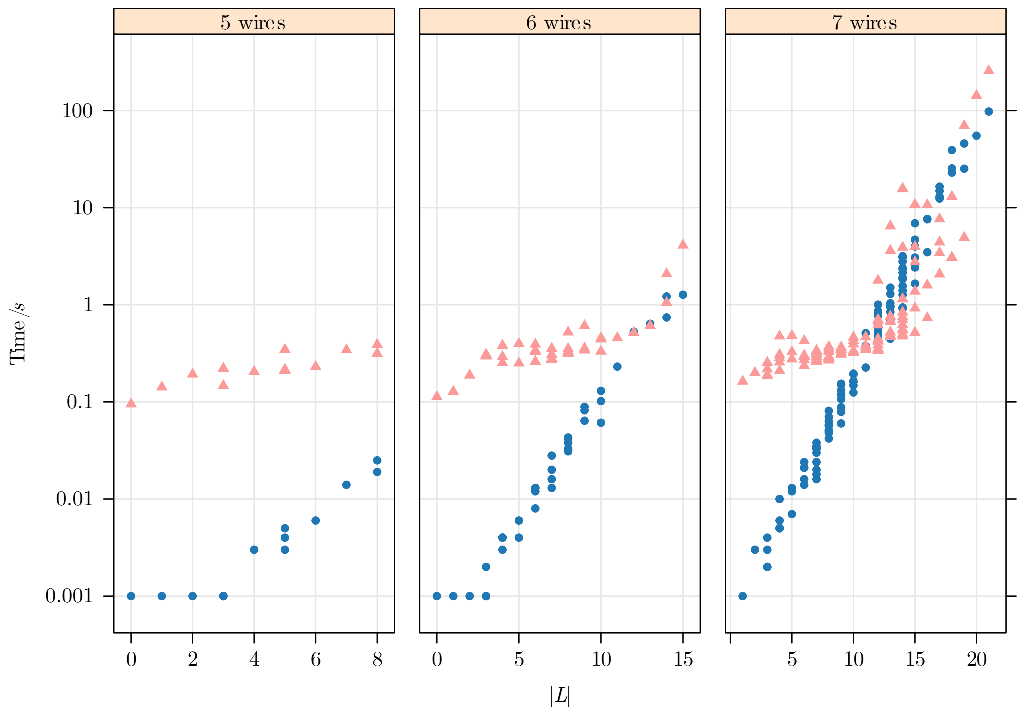

Figure 3 Comparison of our algorithm (blue circles) with the algorithm of Olszewski et al. (red triangles). The elapsed time is plotted on a log-scale.

We obtain the first inequality from the inequality between arithmetic and geometric means, the second one from Bernoulli's inequality, and the third one follows from $1 + x \leq e^x$.

▶ **Theorem 5.** *For a list L of order n,* Tangle-Height Minimization *can be solved in $O((2|L|/n^2 + 1)^{n^2/2}\varphi^n n)$ time.*

4 Experiments

We implemented the algorithm described in Theorem 5 and compared the running time of our implementation with the one of Olszewski et al. [4]. Their code and a database of all possible elementary linking matrices (most of them non-simple) of 5 wires (14 instances), 6 wires (38 instances), and 7 wires (115 instances) are available at `https://gitlab.uni.lu/PCOG`. Both their and our code is implemented in Python.

We ran our experiments on an Intel Core i7-4770K CPU with a clock speed of 3.50 GHz and 16 GB RAM under Windows 10 64bit. We measured the time to create an optimal tangle 5 times and took the arithmetic mean. The results are summarized in Fig. 3. For 8 of the instances with 7 wires, we stopped their algorithm after 2 hours without finding an optimal solution. We removed these instances from the analysis (although our algorithm found a solution all but of them in under four minutes, and the last one in 20 minutes).

On average, our algorithm was considerably faster; its running time was only 2.59% for 5 wires, 28.69% for 6 wires, and 72.85% for 7 wires of the running time of the algorithm by Olszewski et al. Our algorithm is also more space efficient; the memory usage peaked at 1.2 GB, while Olszewski et al. reportedly used up to 1 TB of memory in their experiments.

5 Open Problems

Is it NP-hard to test the feasibility of a given (non-simple) list? Even if feasibility turns out to be NP-hard, can we decide it faster than finding optimal tangles?

We call a list (l_{ij}) *non-separable* if, for any $i<k<j$, $l_{ik} = l_{kj} = 0$ implies $l_{ij} = 0$. Clearly, non-separability is necessary for a list to be feasible. For lists where all entries are even, we conjecture that this is also sufficient (which we have computer-verified for $n \leq 8$).

Acknowledgments. We thank Thomas C. van Dijk for stimulating discussions.

References

1 Sergey Bereg, Alexander Holroyd, Lev Nachmanson, and Sergey Pupyrev. Representing permutations with few moves. *SIAM J. Discrete Math.*, 30(4):1950–1977, 2016. URL: `http://arxiv.org/abs/1508.03674`, `doi:10.1137/15M1036105`.

2 Sergey Bereg, Alexander E. Holroyd, Lev Nachmanson, and Sergey Pupyrev. Drawing permutations with few corners. In Stephen Wismath and Alexander Wolff, editors, *Proc. Int. Symp. Graph Drawing (GD'13)*, volume 8242 of *LNCS*, pages 484–495. Springer, 2013. URL: `http://arxiv.org/abs/1306.4048`, `doi:10.1007/978-3-319-03841-4_42`.

3 Oksana Firman, Philipp Kindermann, Alexander Ravsky, Alexander Wolff, and Johannes Zink. Computing optimal tangles faster. ArXiv report, 2019. URL: `http://arxiv.org/abs/1901.06548`.

4 Maya Olszewski, Jeff Meder, Emmanuel Kieffer, Raphaël Bleuse, Martin Rosalie, Grégoire Danoy, and Pascal Bouvry. Visualizing the template of a chaotic attractor. In Therese Biedl and Andreas Kerren, editors, *Proc. 26th Int. Symp. Graph Drawing & Network Vis. (GD'18)*, volume 11282 of *LNCS*, pages 106–119. Springer, 2018. URL: `http://arxiv.org/abs/1807.11853`, `doi:10.1007/978-3-030-04414-5_8`.

5 Deborah C. Wang. Novel routing schemes for IC layout part I: Two-layer channel routing. In *Proc. 28th ACM/IEEE Design Automation Conf. (DAC'91)*, pages 49–53, 1991. `doi:10.1145/127601.127626`.