MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **78** (2017) 581-594

ISSN 0340 - 6253

An Algebraic Approach to Enumerating Non–Equivalent Double Traces in Graphs

Nino Bašić[a], Drago Bokal[b], Tomas Boothby[c], Jernej Rus[d,*]

[a] *FAMNIT, University of Primorska, Slovenia*
nino.basic@famnit.upr.si

[b] *Faculty of Natural Sciences and Mathematics, University of Maribor, Slovenia*
bokal@uni-mb.si

[c] *Department of Mathematics, Simon Fraser University, Canada*
tboothby@sfu.ca

[d] *Abelium d.o.o., Slovenia*
jernej.rus@gmail.com

(Received December 14, 2016)

Abstract

Recently designed biomolecular approaches to build single chain polypeptide polyhedra as molecular origami nanostructures have risen high interest in various double traces of the underlying graphs of these polyhedra. Double traces are walks that traverse every edge of the graph twice, usually with some additional conditions on traversal direction and vertex neighborhood coverage. Given that double trace properties are intimately related to the efficiency of polypeptide polyhedron construction, enumerating all different possible double traces and analyzing their properties is an important step in the construction. In the paper, we study the automorphism group of double traces and present an algebraic approach to this problem, yielding a branch-and-bound algorithm.

1 Introduction

Gradišar et al. presented a novel self-assembly strategy for polypeptide nanostructure design in 2013 [14]. Their research was already improved by Kočar et al. in 2015, who developed another alternative strategy to design topofolds — nanostructures built from polypeptide arrays of interacting modules that define their topology [16]. Such approaches

*Corresponding author.

are paving the way to a significant breakthrough in the field of protein origami, an area advancing a step ahead of DNA origami, where many researchers have spent the better part of the past decade by folding the molecules into dozens of intricate shapes.

A polyhedron P that is composed from a single polymer chain can be naturally represented by a graph $G(P)$ of the polyhedron. As every edge of $G(P)$ corresponds to a coiled-coil dimer in the self-assembly process, exactly two biomolecular segments are associated with every edge of $G(P)$. Hence, every edge of $G(P)$ is in its biomolecular structure replaced by two copies, resulting in a graph $G'(P)$ obtained from $G(P)$ by replacing every edge with a digon. The graph $G'(P)$ is therefore Eulerian, and its Eulerian walks (i.e., walks that traverse every edge of $G(P)$ precisely twice), called *double traces* of $G(P)$, play a key role in modeling the construction process. Note that the argument shows that every graph admits a double trace.

Double traces with additional properties related to stability of the constructed polyhedra were introduced as a combinatorial model underlying these approaches to polypeptide polyhedra design in [15] and [8]. Stability of the resulting polyhedron depends on two additional properties: one relates to whether in the double trace the neighborhoods of vertices can be split, and the other defines whether the edges of the double trace are traversed twice in the same or in different directions.

To define the first property, let an alternating sequence $W = w_0e_1w_1 \dots w_{2m-1}e_{2m}w_{2m}$, where e_i is an edge between vertices w_{i-1} and w_i, be a double trace — a closed walk which traverses every edge of the graph exactly twice. Note that we always consider the vertex sequence of a double trace with indices taken modulo $2m$. (Since the graph $G(P)$ is simple, so are all our other graphs, except $G'(P)$. Hence, a double trace is completely described by listing the vertices of the corresponding walk and we sometimes write double trace as a sequence consisting only of vertices.) For a set of vertices $N \subseteq N(v)$, a double trace W has a *N-repetition* at vertex v, if whenever W comes to v from a vertex in N it also continues to a vertex in N (clearly $\emptyset$ and $N(v)$ always form a repetition—*trivial* repetition). More formally W has a N repetition at v if the following implication holds:

$$\text{for every } i \in \{0, \dots, 2m-1\}: \text{ if } v = w_i \text{ then } w_{i+1} \in N \text{ if and only if } w_{i-1} \in N.$$

Then, W is a *strong trace* if W is for every vertex v without nontrivial N-repetitions at v. It is a nontrivial result of [8] that every graph admits a strong trace. A weaker concept of *d-stable trace* requires that whenever W has an nonempty N-repetition at some vertex

v, then $|N| > d$ (there is no N-repetition for $1 \leq |N| < d$). Fijavž et al. [8] showed that G admits a d-stable trace if and only if $\delta(G) \geq d$.

For the second property, note that there are precisely two directions to traverse an edge $e = uv$. If the same direction is used both times W traverses e, then e is a *parallel* edge w.r.t. W, otherwise it is an *antiparallel* edge. A double trace W is *parallel*, if all edges of G are parallel w.r.t. W and is *antiparallel*, if all the edges are antiparallel. Interestingly, antiparallel traces appeared (under a different name) two centuries ago in a study of properties of labyrinths by Tarry [23], who observed (in our language) that every connected graph admits an antiparallel double trace. Fijavž et al. extended this by characterizing the graphs that admit an antiparallel strong trace [8], and Rus upgraded the result to characterize graphs that admit an antiparallel d-stable trace [20]. The former characterization can be algorithmically verified using algorithms of [11], but regarding the latter, it is only known that the existence of antiparallel 1-stable traces can be verified using Thomassen's modification of the aforementioned algorithm, as published in [24] and later corrected by Benevant López and Soler Fernández in [2]. Similar modification of algorithm for spanning tree parity problem presented in [12] would work for $d > 1$ as well, rendering the problem "Does there exist an antiparallel d-stable trace in G?" polynomially tractable. It was also analytically proven that some structures (tetrahedron from [14] for example) can not be constructed using only parallel or only antiparallel coiled coil dimers. In this direction Wang et. al. [25, 26] recently investigated which conditions should a subgraph induced by parallel and antiparallel edges fulfill, respectively. Some additional research was also made in [3] and [5].

It is easy to obtain new traces from a given trace: one can change direction of tracing or start at a different vertex. Also, if graph possesses certain symmetries, these may reflect in the trace. Such changes do not alter any properties of the trace, hence we call the resulting traces equivalent, and we are interested in non-equivalent traces, as introduced in [15]:

Definition 1.1 *Two double traces W and W' are called* equivalent *if W' can be obtained from W (i) by reversion of W, (ii) by shifting W, (iii) by applying a permutation on W induced by an automorphism of G, or (iv) using any combination of the previous three operations. If that is not the case, W and W' are* non-equivalent.

Two double traces W and W' are called *different* if their vertex sequences are not the

same. Two different double traces may be equivalent. It is easy to see that equivalence of double traces is an equivalence relation on the *set $\mathcal{T}$ of all different double traces*, and hence on any subset (such as strong traces, d-stable traces etc.). The main contribution of our paper is designing for each of the subsets of interest an algorithm that, for a given graph as an input, outputs precisely one representative of each equivalence class. This representative will be the unique minimal element for the following linear ordering, called *lexicographical ordering* of double traces. We assume that the vertices of G are linearly ordered as $v_0 < v_1 < \cdots < v_{n-1}$, and that v_0, v_1 are adjacent. This linear ordering induces an ordering on the set of double traces as follows:

Definition 1.2 *Given two double traces $W = w_0 \dots w_{2m}$ and $W' = w'_0 \dots w'_{2m}$, W is* lexicographically smaller or equal *to W', denoted $W \leq_{\text{lex}} W'$, if and only if $W = W'$ or the first w_i, which is different from w'_i, is smaller than w'_i.*

As lexicographical order is a linear order, it is clear that any finite set S of double traces has a unique lexicographically smallest member. We call that member the *canonical representative* of S.

For a more detailed treatment of double-trace related definitions we refer the reader to [8]. For other terms and concepts from graph theory not defined here, we refer to [27].

An automorphism $\pi \in \mathrm{Aut}(G)$ acts on $\mathcal{T}$ by mapping a double trace $W = w_0 \dots w_{2m}$ to $\pi(W) = \pi(w_0) \dots \pi(w_{2m})$. Let $\rho : \mathcal{T} \to \mathcal{T}$ be a reversal that maps $W = w_0 \dots w_{2m}$ to $W' = w_{2m} w_{2m-1} \dots w_0$, and, for $i = 0, \dots, 2m$, let σ_i be an i-shift that maps $W = w_0 \dots w_{2m}$ to $W'' = w_i \dots w_{2m+i}$. Note that $\sigma_0 = \sigma_{2m} = \mathrm{id}$. Then the group $\mathrm{Aut}(G)$, the group $R = \{\mathrm{id}, \rho\}$, and the group $S = \{\sigma_i \mid i = 0, \dots, 2m-1\}$ are three groups acting on $\mathcal{T}$ (or its subsets such as strong traces, d-stable traces etc.). Note that groups R and S do not commute and $\langle R, S \rangle$ is a dihedral group of symmetries of a regular $(2m)$-gon, where $|E(G)| = m$. Therefore, the orbits of the direct product $\Gamma = \mathrm{Aut}(G) \times \langle R, S \rangle$ are precisely the equivalence classes of double traces for the relation from Definition 1.1. Hence, a canonical representative of each equivalence class is the lexicographically smallest element of each class. We say that a double trace is *canonical*, if it is the lexicographically smallest element of its orbit, meaning that every element of Γ maps it to a lexicographically larger (or equal) element. Note that to verify canonicity of a particular double trace, it is not enough to check whether the generators of Γ map it to a larger element (we leave finding an example to the reader).

It is easy to see that every canonical double trace starts with v_0v_1 (by assumption, these two vertices are adjacent) and that every double trace is equivalent to at least one canonical double trace. Double traces (not necessary canonical) starting with v_0v_1 are called *simple*. More details on graph automorphisms can be found in [13], but we do conclude this introduction with an example of the action of Γ on $\mathcal{T}$ in the case of the tetrahedron.

Action of Γ on the set $\mathcal{T}$ of all 672 strong traces of a tetrahedron can be graphically presented. Strong traces are vertices of an arc-labeled digraph $D(\mathcal{T}, \Gamma)$ (which consequently has 672 vertices). An arc (i.e., a directed edge) from vertex U to W exists when an element of Γ maps the trace U to the trace W; the label on that arc is the element of Γ that maps U to W. Note that all (strongly) connected components of that digraph correspond to orbits under the action of Γ. In the case of a tetrahedron, Γ partitions $\mathcal{T}$ into 3 orbits of orders 288, 288, and 96. In other words, the digraph is a union of three complete digraphs of orders 288, 288, and 96. This fact coincides with the results of Table 1 (presented in Section 3). Subgroups $\mathrm{Aut}(G)$, R, and S partition $\mathcal{T}$ into 28 orbits of order 24, 336 orbits of order 2, and 56 orbits of order 12, respectively. The digraph $D(\mathcal{T}, \Gamma)$ defined above is a very dense one. A more sparse digraph can be defined if a set of generators of Γ is used instead of the whole group. Such digraph is more convenient for making observations. The subgroup S is generated by the element σ_1 and the subgroup R is generated by ρ. In the case of the tetrahedron $\mathrm{Aut}(G) \cong S_4$. It is known that every symmetric group on a finite set is 2-generated, i.e., a minimal set of generators has cardinality at most 2. Let the vertices of the tetrahedron be labeled with numbers 1, 2, 3 and 4. Then $\mathrm{Aut}(G) = \langle \alpha_1, \alpha_2 \rangle$ where $\alpha_1 = (1\,2)$ and $\alpha_2 = (1\,2\,3\,4)$. Therefore, $\Gamma = \langle \alpha_1, \alpha_2, \sigma_1, \rho \rangle$. We may encode traces of the tetrahedron as words of length 13 which comprise of digits $1, \ldots, 4$. The reader can verify that $W = 1342312412341$ is a strong trace in the tetrahedron. Figure 1 shows a small out-neighborhood of the vertex 1342312412341 in the digraph $D(\mathcal{T}, (\alpha_1, \alpha_2, \sigma_1, \rho))$ that is described above.

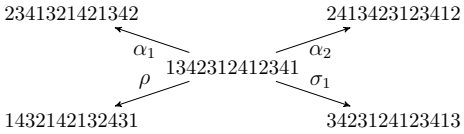

Figure 1. The out-neighborhood of the vertex 1342312412341 in the digraph $D(\mathcal{T}, (\alpha_1, \alpha_2, \sigma_1, \rho))$.

This is (to our knowledge) the first analysis of the automorphism group of a double trace. We proceed as follows. In Section 2, we use the automorphism group to devise a branch-and-bound algorithm that outputs each canonical strong double trace of G precisely once. The main idea of the algorithm is avoiding isomorphic double traces by extending minimal forms. Such an idea was first presented in [7] and [19] where it was called the orderly generation. It is not difficult to see that with minor adjustments, this algorithm can enumerate other varieties of double traces, such as d-stable traces, parallel double traces, or antiparallel double traces. We conclude, in Section 3, with some numerical results that reveal possible varieties in designing polyhedral polypeptide nanostructures.

2 Enumerating strong traces with branch-and-bound strategy

In this section we assume that the n vertices of some arbitrary, but fixed, connected graph G with m edges are linearly ordered as $v_0 < v_1 < \cdots < v_{n-1}$, and that v_0, v_1 are adjacent. Therefore every canonical double trace of G starts with v_0v_1. We denote the automorphism group of double traces in graph G with Γ. To make the arguments more transparent, let W and W' from now on be two different double traces. We first give some additional observations.

Definition 2.1 *Let $W = w_0 \dots w_{2m}$ be a double trace of a graph G. An* initial segment $\mathrm{init}(W)$ *of W is the shortest continuous subsequence of W such that* $\mathrm{init}(W)$ *starts in w_0 and contains all the vertices from $V(G)$.*

Definition 2.2 *Let $W = w_0 \dots w_{2m}$ be a double trace. Then an* i-initial segment *of W, denoted W_i, is a subsequence of first i vertices in W, i.e., $W_i = w_0 \dots w_{i-1}$.*

Definition 2.3 *A double trace W is* i-canonical *if for every $\pi \in \Gamma$, the relation $W_i \leq_{\mathrm{lex}} \pi(W_i)$ holds.*

Lemma 2.4 *A double trace W of length $2m$ is canonical if and only if W is i-canonical for all i, $1 \leq i \leq 2m$.*

Proof. Let a double trace $W = w_0 \dots w_{2m}$ be i-canonical for every i, $1 \leq i \leq 2m$. Suppose that W is not canonical. Since W is also $2m$-canonical, $W_{2m} \leq_{\mathrm{lex}} \pi(W_{2m})$ for

every $\pi \in \Gamma$. Therefore, if W is not canonical, there exists an automorphism $\pi \in \Gamma$, for which $w_i = \pi(w_i)$ for every $0 \leq i \leq 2m-1$ and $\pi(w_{2m}) < w_{2m}$. This is absurd, since $w_0 = w_{2m}$.

Let now W be a canonical double trace of length $2m$. Suppose that for some i, $1 \leq i \leq 2m$, W is not i-canonical. Then there exists $\pi \in \Gamma$, such that $\pi(W_i) <_{\text{lex}} W_i$. Because W is canonical, it follows that $W \leq_{\text{lex}} \gamma(W)$ for every $\gamma \in \Gamma$. Therefore, $W \leq_{\text{lex}} \pi(W)$. From Definition 1.2 it is clear that $W \leq_{\text{lex}} W'$ implies $W_i \leq_{\text{lex}} W'_i$ for every i. Therefore, $W_i \leq_{\text{lex}} \pi(W_i)$, a contradiction. ■

We first explain the auxiliary algorithms used in the main Algoritm 4. If G is a graph with m edges and $p \leq 2m$, then vertex sequence $W_p = w_0 \dots w_{p-1}$ is a *partial double trace* if there exists a double trace W of G for which W_p is its p-initial segment. Analogously, we define partial double trace for other varieties of double traces (strong and d-stable traces). Let W_p be a partial double trace of length p. Let $\mathcal{W}_p$ represent all double traces in G for which their p-initial segment is equal to W_p. We say that W_p is lexicographically smaller than or equal to a different partial double trace W'_p if $W_p \leq_{\text{lex}} W'_p$ (i.e., if $W_p = W'_p$ or the first w_i, which is different from w'_i, is smaller than w'_i). We say that W_p is canonical if at least one double trace from $\mathcal{W}_p$ is canonical. Stabilizer of a partial double trace W_p is defined as the subset of all automorphisms in Γ which map at least one double trace from $\mathcal{W}_p$ back to (not necessary the same) double trace from $\mathcal{W}_p$.

Feasible neighbor of w_{p-1} in a partial double trace W_p is any vertex $v \in N(w_{p-1})$ for which it holds that $W_{p+1} = w_0 \dots w_{p-1}v$ (that is obtained from W_p by adding v) is also a partial double trace. This can be analogously defined for partial strong traces and d-stable traces, where we have to be careful that v does not cause any new nontrivial repetition of excessive order. For antiparallel or parallel double traces we additionally forbid vertices that cause parallel or antiparallel edges in partial double trace, respectively.

Algorithm 1 loops through all the feasible neighbors of the last vertex w_{p-1} in a partial double trace $W_p = w_0 \dots w_{p-1}$ and checks which of them, if added to W_p (to obtain partial double trace W_{p+1}), will retain a canonical partial double trace. Partial double traces obtained in this procedure are added to a queue Q.

At each step we use the automorphism group of double traces Γ in order to eliminate all partial double traces that would not lead to a construction of a canonical double trace. We achieve that by considering only the lexicographically smallest representative

Algorithm 1 EXTEND FEASIBLY

Input: a partial double trace $W_p = w_0 \dots w_{p-1}$,
$A \subseteq \Gamma$,
a queue Q of partial double traces.
Output: updated queue Q
$V' =$ FEASIBLE NEIGHBORS(w_{p-1})
$V'' =$ CANONICAL EXTENSION(V', W_p, A)
for $v \in V''$ **do**
$W_{p+1} = w_1 \dots w_{p-1} v$
$A_v =$ PRUNE(A, W_{p+1})
if W_{p+1} is canonical partial double trace **then**
append (W_{p+1}, A_v) to Q

of each orbit of the automorphism group. Simultaneously, we would like to fix vertices that are already in a partial double trace, since we have already checked it for canonicity. Therefore, Algorithm 2 returns the automorphisms that are in the stabilizer of partial double trace W_{p+1} (in each step only the last position p has to be checked). Note that until the double trace is not completed, we can not determine for each automorphism from Γ if the automorphism lies in the stabilizer of the partial double trace. The problem is in shifting, since we can not always deteremine how all first p places of the shifted partial double trace look like. Therefore, we do not discard such an automorphism at this point.

Algorithm 2 PRUNE

Input: a set of automorphisms A of double traces,
partial double trace W_p.
Output: pruned set of automorphisms A' of double traces.
$A' = \emptyset$
for $\pi \in A$ **do**
if (π in stabilizer of partial double trace W_p) or (it can not be determined if π is in a stabilizer of W_p) **then**
append π to A'
return A'

Algorithm 3 loops through all the feasible neighbors of the last vertex w_{p-1}, denoted $V \subseteq N(w_{p-1})$, of a partial double trace $W_p = w_0 \dots w_{p-1}$. For every $v \in V$ it constructs a new partial double trace $W_{p+1} = w_0 \dots w_{p-1} v$. Denote a set of those new partial double traces with $\mathcal{W}_{p+1}$ and analyses orbits of $\mathrm{Aut}(G) \cap A$ (no shifts and reverses are allowed, therefore each orbit contains an even smaller number of partial double traces) acting on the set of these new partial double traces. Then for every such orbit O this algorithm selects a vertex $v \in V$ for which the partial double trace $W_p = w_0 \dots w_{p-1} v$ is the lexicographically

smallest among all partial double traces in O. Note that in practice, this algorithm should only check the position p since Algorithm 2 ensures that for every $\pi \in \mathrm{Aut}(G) \cap A$ the vertices $w_0, \ldots, w_{p-1}$ are fixed.

Algorithm 3 CANONICAL EXTENSION

Input: a partial double trace $W_p = w_0 \ldots w_{p-1}$,
a set of feasible neighbors $V \subseteq N(w_{p-1})$,
a set of automorphisms $A \subseteq \Gamma$.
Output: the set $V' \subseteq V$ that for each orbit O of $\mathrm{Aut}(G) \cap A$ acting on $\mathcal{W}_{p+1}$ contains a vertex v for which $W_{p+1} = w_0 \ldots w_{p-1} v$ is the lexicographically smallest partial double trace of orbit O.

```
if A = ∅ or A = {id} then
    return V
V' = ∅
for v ∈ V do
    V'' = {v}
    v' = v
    for π ∈ Aut(G) ∩ A do
        append π(v) v V''
        if w_0 ... w_{p-1} π(v) <_lex w_0 ... w_{p-1} v' then
            v' = π(v)
    append v' to V'
    V = V \ V''
return V'
```

We now present the main Algorithm 4, which enumerates strong traces for an arbitrary graph. In the rest of the section, we prove the correctness of Algorithm 4.

Algorithm 4 ENUMERATE STRONG TRACES

Input: a graph G with m edges,
the automorphism group Γ of double traces of G.
Output: the list L of all canonical non-equivalent double traces.

```
W_1 = v_0 v_1
A = Aut(G)
A = PRUNE(A, W_1)
Q = {(W_1, A)}
while Q not empty do
    (W, A) = head of Q
    remove (W, A) from Q
    if |W| = 2m then
        add W to L
    else
        EXTEND FEASIBLY(W, A, Q)
return L
```

Theorem 2.5 *Let W be a double trace, which was given as an output of Algorithm 4. Then W is canonical.*

Proof. Each partial double trace W_p, which was added to Q by the Algorithm 1 is i-canonical for every $1 \leq i \leq p$. For the special case of $p = 2m$ then follows, that a double trace added by an Algorithm 1 to Q (and is also returned in set L by the Algorithm 4) is i-canonical, for every $1 \leq i \leq 2m$. By the Lemma 2.4 such W is canonical. ■

All double traces, which were given as an output of Algorithm 4 are non-equivalent. Otherwise some equivalence class of double traces would include two canonical double traces, which contradicts the definition of canonical double trace.

Theorem 2.6 *Let W be a canonical double trace. Then W is given as an output of Algorithm 4.*

Proof. Suppose the contrary. Let $W = w_0 \dots w_{2m}$ be a canonical double trace which is not given as an output of Algorithm 4. By observations made in Section 1, W starts with v_0v_1. There exists the largest integer i (at least $i = 1$, if no other) such that W_i is the i-initial segment of some canonical double trace which is an output of Algorithm 4. Let $\mathcal{W}_i$ be the set of all (canonical) double traces which are given as an output of Algorithm 4 and have W_i as their i-initial segment. Let V_{i+1} be the set of vertices that lie at the $(i+1)$-th position in (canonical) double traces from $\mathcal{W}_i$. It follows that $w_{i+1} \notin V_{i+1}$. Since W is a double trace, w_{i+1} was in Algorithm 4 (in Algorithm 3, to be more precise) part of feasible neighbors of w_i for every double trace from $\mathcal{W}_i$. Since it was never added, it follows that in the same orbit of $\mathrm{Aut}(G) \subseteq \Gamma$ that contains W, there also lies another (lexicographically smaller) double trace $W' \in \mathcal{W}_i$. That contradicts the fact that W is canonical. ■

We presented an algorithm which enumerates all non-equivalent double traces of a graph G. To enumerate only strong traces or only d-stable traces of the graph G, we just have to restrict the conditions on feasible neighbors. For strong traces, we need to forbid non-trivial repetitions. For d-stable traces, we need to forbid repetitions of order at most d. It is also possible to enumerate parallel or antiparallel double traces. For some edge $e = uv$, which is already present in a partial double trace W_i, we need to store the information on the direction of traversal (which can be either from u to v or vice versa). This imposes additional restrictions on the set of feasible neighbors of vertices u and v.

3 Concluding remarks and numerical results

We conclude with some numerical results. In Tables 1, 2 and 3, we present enumerations of non-equivalent strong traces for platonic solids, prisms, and some other interesting polyhedra which could be the next candidates to be constructed from coiled-coil-forming segments. Note that d, n, m, ST, aST and pST stand for the degree of the graph (if the graph is regular), the number of its vertices and edges, the number and the CPU time used to enumerate strong traces, the number and the CPU time used to enumerate antiparallel strong traces, and number and the CPU time used to enumerate parallel strong traces in Tables 1, 2 and 3, respectively. Note that the listed CPU times are measured in seconds. For comparison, CPU times for a brute-force search algorithm (first generating all double traces and then checking for each of them if it is canonical) are also given in brackets. In addition to the number of strong traces, the algorithm for every strong trace also returns its vertex sequence. Therefore, it can be used for a thorough analysis of some properties that nanostructures, which self-assemble from these strong traces, would have. Further, this analysis helps to select a strong trace with the maximal probability of giving rise to a stable nanostructure of desired shape.

				ST		pST	
graph	d	n	m	#	CPU time	#	CPU time
tetrahedron	3	4	6	3	0.003 (0.003)	0	n/a
cube	3	8	12	40	0.009 (0.012)	0	n/a
octahedron	4	6	12	21479	1.860 (4.470)	262	0.056 (0.111)
dodecahedron	3	20	30	2532008	1962.620 (4943.810)	0	n/a

Table 1. Number of strong traces and parallel strong traces for platonic solids.

All the calculations were made with Algorithm 4 using the open source mathematical software SageMath and computational resources at SageMathCloud [22] (8GB of RAM, 4 CPU cores). It was observed in [8], that a graph G admits a parallel strong trace if and only if G is Eulerian, and that G admits an antiparallel strong trace if and only if there exists a spanning tree T of G with the property that every component of the co-tree $G - E(T)$ is even. Therefore, we omit the information about antiparallel and parallel strong traces for graphs not admitting them. Some of these calculations were already presented in [14] and [15].

Another possible approach to strong trace construction exploits the observation that

				ST		aST	
graph	d	n	m	#	CPU time	#	CPU time
$K_2 \square C_3$	3	6	9	25	0.004 (0.004)	2	0.004 (0.004)
$K_2 \square C_4$	3	8	12	40	0.010 (0.012)	0	n/a
$K_2 \square C_5$	3	10	15	634	0.047 (0.059)	10	0.005 (0.005)
$K_2 \square C_6$	3	12	18	3604	0.377 (0.576)	0	n/a
$K_2 \square C_7$	3	14	21	21925	3.210 (4.060)	76	0.024 (0.035)
$K_2 \square C_8$	3	16	24	134008	26.060 (32.920)	0	n/a
$K_2 \square C_9$	3	18	27	833685	210.760 (285.460)	536	0.332 (0.507)
$K_2 \square C_{10}$	3	20	30	5212520	1719.770 (2258.790)	0	n/a

Table 2. Number of strong traces and antiparallel strong traces for prisms.

				ST		aST	
graph	d	n	m	#	CPU time	#	CPU time
4-pyramid	n/a	5	8	52	0.003 (0.004)	4	0.003 (0.003)
3-bipyramid	n/a	5	9	470	0.009 (0.018)	0	n/a

Table 3. Number of strong traces and antiparallel strong traces in 4-pyramid and 3-bipyramid.

a strong trace can be nicely drawn on a surface in which the given graph is embedded. This surface can be cut along certain edges which results in one or more surfaces with boundary. Each of the resulting surfaces with boundary carries a part of the information about the strong trace. The strong trace can be reconstructed by gluing those smaller pieces back together. This topological approach will be elaborated in [1].

Acknowledgments: The authors would like to thank Gunnar Brinkmann for many helpful suggestions and Anders Skovgaard Knudsen who independently calculated the number of strong traces in platonic solids and shared the results for comparison. This research was supported in part by Slovenian Research Agency under research grants P1-0294, P1-0297, L7-5459 and L7-5554.

References

[1] N. Bašić, D. Bokal, T. Pisanski, J. Rus, Graph embeddings yield natural strong trace realizations, in preparation.

[2] E. Benevant López, D. Soler Fernández, Searching for a strong double tracing in a graph, *Top* **6** (1998) 123–138.

[3] H. J. Broersma, F. Göbel, k-Traversable graphs, *Ars Comb.* **29** (1990) 141–153.

[4] R. B. Eggleton, D. K. Skilton, Double tracings of graphs, *Ars Comb.* **17** (1984) 307–323.

[5] J. A. Ellis-Monaghan, A. McDowell, I. Moffatt, G. Pangborn, DNA origami and the complexity of Eulerian circuits with turning costs, *Nat. Comput.* **14** (2015) 491–503.

[6] L. Euler, Solutio problematis ad geometriam situs pertinentis, *Commentarii Academiae Scientiarum Imperialis Petropolitanae* **8** (1741) 128–140.

[7] I. A. Faradžev, Constructive enumeration of combinatorial objects, *Colloques internationaux C.N.R.S. No260 - Problémes Combinatoires et Théorie des Graphes*, Orsay, 1976, pp. 131–135.

[8] G. Fijavž, T. Pisanski, J. Rus, Strong traces model of self–assembly polypeptide structures, *MATCH Commun. Math. Comput. Chem.* **71** (2014) 199–212.

[9] H. Fleischner (Ed.), *Eulerian Graphs and Related Topics. Vol. 1*, North–Holland, Amsterdam, 1990.

[10] H. Fleischner (Ed.), *Eulerian Graphs and Related Topics. Vol. 2*, North–Holland, Amsterdam, 1991.

[11] M. L. Furst, J. L. Gross, L. A. McGeoch, Finding a maximum–genus graph imbedding, *J. Assoc. Comput. Mach.* **35** (1988) 523–534.

[12] H. N. Gabow, M. Stallman, An augmenting path algorithm for linear matroid parity, *Combinatorica* **6** (1986) 123–150.

[13] C. Godsil, G. Royle, *Algebraic Graph Theory*, Springer, New York, 2001.

[14] H. Gradisar, S. Božič, T. Doles, D. Vengust, I. Hafner Bratkovič, A. Mertelj, B. Webb, A. Šali, S. Klavžar, R. Jerala, Design of a single–chain polypeptide tetrahedron assembled from coiled–coil segments, *Nat. Chem. Biol.* **9** (2013) 362–366.

[15] S. Klavžar, J. Rus, Stable traces as a model for self-assembly of polypeptide nanoscale polyhedrons, *MATCH Commun. Math. Comput. Chem.* **70** (2013) 317–330.

[16] V. Kočar, S. Božič Abram, T. Doles, N. Bašić, H. Gradišar, T. Pisanski, R. Jerala, Topofold, the designed modular biomolecular folds: polypeptide-based molecular origami nanostructures following the footsteps of dna, *WIREs Nanomed. Nanobiotech.* **7** (2015) 218–237.

[17] B. D. McKay, A. Piperno, Practical Graph Isomorphism, II, *J. Symb. Comput.* **60** (2014) 94–112.

[18] T. Pisanski, A. Žitnik, Representing graphs and maps, in: L. W. Beineke, R. J. Wilson (Eds.), *Topics in Topological Graph Theory*, Cambridge Univ. Press, Cambridge, 2009, pp. 151–180.

[19] R. C. Read, Every one a winner, *Annals Discr. Math.* **2** (1978) 107–120.

[20] J. Rus, Antiparallel d-stable traces and a stronger version of Ore problem, *J. Math. Biol.*, in press.

[21] G. Sabidussi, Tracing graphs without backtracking, in: R. Henn, P. Kall, B. Korte, O. Krafft, W. Oettli, K. Ritter, J. Rosenmüller, N. Schmitz, H. Schubert, W. Vogel (Eds.), *Methods of Operations Research XXV, Part 1*, Univ. Heidelberg, Heidelberg, 1977, pp. 314–332.

[22] The Sage Developers, *Sage Mathematics Software (Version 7.4.(2016-10-18)*, 2016, `http://www.sagemath.org`.

[23] G. Tarry, Le problème des labyrinthes, *Nouv. Ann.* **3** (1895) 187–190.

[24] C. Thomassen, Bidirectional retracting-free double tracings and upper embeddability of graphs, *J. Comb. Theory Ser. B* **50** (1990) 198–207.

[25] J. Wang, G. Hu, M. Ji, Almost parallel strong trace model of self-assembly polypeptide nanostructure, *MATCH Commun. Math. Comput. Chem.* **77** (2017) 783–798.

[26] J. Wang, X. Jin, M. Ji, On the existence of F-strong trace of a graph when F induces a forest, *MATCH Commun. Math. Comput. Chem.* **77** (2017) 799–812.

[27] D. B. West, *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River, 1996.