Linear Time Canonicalization and Enumeration of Non-Isomorphic 1-Face Embeddings

Marc Hellmuth [*] Anders S. Knudsen [†] Michal Kotrbčík [‡] Daniel Merkle [§]
Nikolai Nøjgaard [¶]

Abstract

Antiparallel strong traces (ASTs) are a type of walks in graphs which use every edge exactly twice. They correspond to 1-face embeddings in orientable surfaces and can be used to design self-assembling protein or DNA strands. Based on a novel canonical form invariant for ASTs, *gap vector*, we provide a linear-time isomorphism test for ASTs and thus, also for orientable 1-face embeddings of graphs. Using the canonical form, we develop an algorithm for enumerating all pairwise non-isomorphic 1-face embeddings of graphs. We compare our algorithm with an independent implementation of a recent algebraic approach (Bašić *et al.*, MATCH Commun. Math. Comput. Chem. 78 (3), 2017) on large data sets. Our results yield the first large-scale enumeration of non-isomorphic embeddings and investigation of their properties.

1 Introduction.

Two embeddings of a graph are isomorphic if their sets of faces coincide, where two faces are considered the same either if their boundary walks are the same, or if one is the reverse of the other. Enumeration of embeddings, both with and without regard to isomorphism, is a classical topic which attracted a significant amount of attention, see for example [15, 16, 35, 36]. However, the enumeration of embeddings is difficult and very little progress has been made in enumerating pairwise non-isomorphic embeddings. In particular, the precise number of pairwise non-isomorphic embeddings is known only for a few very small graphs. The three main contributions of this paper are: (i) linear-time isomorphism test for 1-face embeddings based on a novel embedding invariant; (ii) experimental evaluation of the resulting enumeration algorithm in comparison with a previous algebraic approach [4]; and (iii) a large amount of empirical information obtained about embeddings of graphs in our test sets, which include all connected graphs with at most nine vertices, all cubic graphs with at most 22 vertices, and all trees with 15 to 20 vertices. This is the first time that such data is available for diverse sets of graph classes consisting of more than only a few graphs. Even in these sets, the number of pairwise non-isomorphic 1-face embeddings of a sibgle graph can be fairly large ($\gg 10^6$). All implemented algorithms, as well as the computed results, are freely available [1, 2]. All designed algorithms are nonspecific in the sense that the same algorithm can be used for any graph class, and hence our implementations provide a workbench for possible further experimental investigations.

1.1 Background.

Our work falls into the scope of topological graph theory, which is concerned with the study of embeddings of graphs on surfaces. Here, an embedding is, loosely speaking, a representation of the graph on a surface, such as a torus, in such a way that the curves representing the edges do not cross.

Central to this area of graph theory are the *minimum* and *maximum genus* of a graph corresponding to the minimum (resp. maximum) genus of an orientable surface into which the graph can be cellularly embedded. The minimum and maximum genus and, more generally, the *genus distribution* - the number of embeddings of G into each surface into which it can be embedded, are arguably the most fundamental topological invariants of a graph and a large body of research is devoted to them, see for example [13, 15, 16, 19, 22, 31, 37].

Since determining the genus distributions of nontrivial graph families exactly seems to be well out of reach of the existing methods, currently the focus is on the type of the embedding distributions. It is conjectured that all embedding distributions are unimodal [18] and it is suspected that they strongly concentrate close to the maximum genus (see Table 1 for an example). This may not come as surprising since there is a polynomial-time algorithm for maximum genus, while the minimum genus problem is NP-hard. By the Euler formula (Eq. (2.1)), maximizing the genus corre-

[*]Dpt. of Mathematics and Computer Science, University of Greifswald, Walther-Rathenau-Strasse 47, D-17487 Greifswald, Germany. mhellmuth@mailbox.org

[†]Department of Mathematics and Computer Science, University of Southern Denmark, Denmark. antersenator@gmail.com

[‡]The University of Queensland, Department of Mathematics, QLD 4072, Australia. m.kotrbcik@gmail.com

[§]Department of Mathematics and Computer Science, University of Southern Denmark, Denmark. daniel@imada.sdu.dk

[¶]Dpt. of Mathematics and Computer Science, University of Greifswald, Walther- Rathenau-Strasse 47, D-17487 Greifswald, Germany; Department of Mathematics and Computer Science, University of Southern Denmark, Denmark. nojgaard@imada.sdu.dk

sponds to minimizing the number of faces. In the extremal cases when the graphs have an embedding with 1 or 2 faces (depending on the parity of $|E|-|V|$), the graphs are called *upper embeddable*. Besides the indications that a large portion of embeddings is very close to the maximum genus, the prominence of 1-face embeddings is further signified by the fact that every 4-edge-connected graph is upper-embeddable. Therefore, every 4-edge-connected graph with $|E|-|V|$ odd has a 1-face embedding. There is essentially one known method of constructing 1-face embeddings, which is based on decomposing the graph into a spanning tree T and a partition of $E(G)-E(T)$ into pairs of adjacent edges. By inserting these pairs of adjacent edges into different positions in the rotation system it is possible to show that every graph has exponentially-many maximum-genus embeddings [30]. However, it is not known how tight this bound is and there are no results on the number of pairwise non-isomorphic maximum-genus embeddings beyond a few very small examples.

1.2 Our Contribution. In this work, we propose three novel invariants collectively called *gap representations*, namely *vertex, edge*, and *bio gap*. For each element of a trace (i.e. the boundary walk of the single face of the embedding), the gap vectors contain the distance to the next (for vertex gap and edge gap), resp. the nearest (bio gap) occurrence of this element, see Section 3 for details. We propose a canonical form CanonGAP of the vertex gap vector obtained as the lexicographically minimal vector in the equivalence class of all shifts and reversals of the gap vector. We prove that two strong traces, or equivalently, two 1-face embeddings, are isomorphic if and only if they have the same CanonGAP. We provide a linear-time algorithm computing CanonGAP of an arbitrary graph and based on it we design a new algorithm, GapEST (**GapE**numeration of **S**trong **T**races), for the enumeration of pairwise non-isomorphic orientable 1-face embeddings of an arbitrary graph. For comparison and to determine the suitability of these algorithms, we also provide an independent implementation of a recent algorithm from [4], which we call AEST (**A**lgebraic **E**numeration of **S**trong **T**races). Gap representations can additionally be used for early pruning the search space to speed-up either of the algorithms, or when searching only for traces with particular properties.

In addition to the latter results, our main contributions are experimental evaluation and data collection. For the evaluation, we compare AEST and GapEST algorithms, investigate hardness of instances depending on various graph parameters, and explore the feasibility of gap representations for the speed-up via pruning.

To evaluate the algorithms, we test both implementations and three pruning strategies based on gaps on several graph classes. As a baseline have chosen the sets of all connected graphs with at most nine vertices. Moreover, due to their suitable size and their interesting topological features we have additionally chosen all cubic graphs with at most 22 vertices, and all trees with 15 to 20 vertices. With time limit 30 seconds per graph, we solve $56,4\%$ of feasible instances on 9 vertices and all the cubic and tree instances.

In the data collection direction, we are driven by the lack of understanding of behavior of embedding parameters. Therefore, we provide a very large amount of information about embeddings and their properties in general graph classes, and the dependency of the properties on graph parameters such as girth and connectivity. While from the theoretical point of view it is well known that these parameters affect embeddability, up to this point it was not known how significant these effects are and whether they can be observed on scale. It should be noted that the number of pairwise non-isomorphic 1-face embeddings of a single graph can be rather high ($\gg 10^6$) even for graphs on 8 or 9 vertices (cf. Table 1). In total, we successfully solve and obtain the exact number of pairwise non-isomorphic orientable 1-face embeddings for over 8 millions graphs. This is the first time that such data is available for diverse sets of graph classes consisting of more than only a small number of graphs. We expect that the data collected will be of value to theoreticans for deepening their understanding of the relationships among the parameters, forming their intuition, and suggesting areas for further study. Since the gap representations seems to be quite interesting on their own, to a certain extent we also investigate their properties and collect empirical information about their distributions.

1.3 Potential Applications. Recently, Gradisar et al. [12] and Kovcar et al. [25] devised methods for building self-assembling protein and DNA nanostructure polyhedra from linear chains. It turns out that under a natural stability requirements, the methods are feasible exactly when the linear chain corresponds to a strong double trace of the desired polyhedra, see also [10] for details. Motivated by this biological development, theoretical properties of traces were revisited [3, 11, 24, 32] and Bašić et al. [4] proposed a branch-and-bound algorithm computing all pairwise non-isomorphic strong traces. The algorithm is based on pruning of the search tree using the automorphisms of the graph and can also enumerate and compute all Antiparallel Strong Traces (ASTs). Since the assembly of a protein or DNA is constrained by its chemical properties, besides listing all strong traces or ASTs it is desirable to be able to also restrict the set of all ASTs of the polyhedra and to filter it according to additional properties. The likelihood of two segments to bind in the biochemical setting depends on their proximity. It is conceivable that the proximity might be reflected by the biological gap representation, which we introduce in Section 3. Therefore, protein or DNA sequences might be more

Genus:	♯ embeddings:	Genus:	♯ embeddings:	Genus:	♯ embeddings:
0	0	3	3 746 107 320	6	158 500 382 165 280
1	240	4	594 836 922 960	7	178 457 399 105 280
2	3 396	5	20 761 712 301 960	8	0

Table 1: The genus distribution of K_7 [14]. The minimum and maximum genus of K_7 is 1 and 7, respectively. The embeddings of genera $1-4$ constitute less than 0.2% of all embeddings, while embeddings of maximum genus and one less constitute over 97% of all embeddings. The embeddings of K_7 with genus 7 have 1 face by Euler Formula (Eq. (2.1)).

likely to self-assemble for a graph G if $\max(\texttt{BioGAP}(t))$ is not too large for the BioGAP-optimal strong trace t, but this would have to be tested by biological experiments. Our implementations allow to filter and investigate any set of candidate graphs according to BioGAP. The graphs in our evaluation are slightly larger than the maximum size of chains that are possible to be assembled now and mostly likely will in the foreseeable future. Therefore, our tools provide readily available methods to enumerate and investigate candidate graphs of appropriate size. The runtime costs of such investigations are in at most days of single-core equivalent of a typical desktop machine for completely solving a single-graph instance, or in hundreds of days to solve majority of a large set of instances. Note that such costs are at least partly unavoidable since the number of all embeddings or the number of ASTs can be prohibitive from the computational perspective even for very modest graph sizes. Therefore, searching only for specific embeddings might be desirable or necessary. Gap vectors, and BioGAP in particular, are natural candidates for partitioning search and solution spaces to, at least partly, mitigate these difficulties.

1.4 AEST algorithm.

AEST algorithm [4] uses a branch-and-bound strategy to prune partial strong traces using the automorphism group of the graph. More precisely, it generates partial traces and uses a set of automorphisms to decide whether the partial trace can be lexicographically minimum in its equivalence class. The lexicographically minimum strong trace is then the canonical representative of its class. The set of automorphisms used starts as the set of all automorphisms and is pruned as the partial trace is being extended. Note that the AEST algorithm, in principle, can enumerate also strong traces which are not antiparallel (see the next section for the precise definitions), while our work focuses only on ASTs. The paper [4] illustrates the performance of the AEST algorithm on a small number of cases; in Table 2 we summarize the cases focusing on ASTs. Since no implementation is publicly available, we reimplemented the AEST algorithm and compare it with our algorithm in Section 5.

Graph:	Y_3	Y_5	Y_7	Y_9	Py_4
Run time (s):	0.005	0.006	0.024	0.43	0.008
♯ASTs:	2	10	76	536	4

Table 2: Runtimes and numbers of ASTs produced by AEST algorithm. The graph Y_i is the prism on $2i$ vertices, that is, two cycles of length i with the corresponding vertices joined by an edge. Py_4 is the graph obtained by joining a new vertex to each vertex of a 4-cycle. The running times corresponds to runs in the SageMathCloud, see [4] for details.

2 Preliminaries.

This section gives a brief overview of the used main concepts and definitions. For further and detailed discussion of topological graph theory, we refer the reader to [19] and [28].

All the graphs in this paper are simple, finite, and undirected. We use standard graph-theoretic terminology, in particular, a graph $G=(V,E)$ has a vertex set V and an edge set E and the sizes of these sets are denoted by n and m, respectively.

For a given graph G, an *embedding* Π of G is (informally) a representation of G on a surface S in such a way that the vertices correspond to different points of S and the edges of G do not cross. We deal only with orientable surfaces; an orientation is a choice of the preferred direction on the surface, either clockwise or counterclockwise. It is known that any orientable surface is homeomorphic to a generalized torus S_g, or equivalently, to a sphere with g handles attached. For S_g, the number g is the *genus* of the surface. It is customary to work only with cellular embeddings where each face is an open disc. We do not need formal topological definition of an embedding since it is known that cellular embeddings in orientable surfaces (with a chosen orientation) are in 1-to-1 correspondence with rotation systems of G as follows. For a vertex v, a *rotation* Π_v on v is a cyclic permutation of edges incident with v. A *rotation system* of G is a set of rotations, one for each vertex of G. For a given embedding $\Pi(G)$, the set of faces of Π is denoted by $F(\Pi)$. Since each face is a cyclic walk $F=(v_1v_2\ldots v_k)$, the starting point bears no significance and all the rotations $(v_2\ldots v_kv_1)$, $(v_3\ldots v_kv_1v_2)$, $\ldots$ corresponding to F are identical.

Clearly, every graph has $\Pi_v(\deg_v-1)!$ distinct rotation systems and thus also embeddings. We sometimes call this *the number of all combinatorial embeddings* when we want to stress that we are comparing this number to the number of (pairwise non-isomorphic) 1-face embeddings.

For a graph G cellularly embedded in S_g with $|F|$ faces, the Euler formula

$$|V| - |E| + |F| = 2 - 2g \tag{2.1}$$

implies that embeddings of G minimizing the number of faces maximize the genus of the surface. In the rest of the paper, the adjectives cellular and orientable will usually be omitted, since all our embeddings are cellular and all the surfaces are orientable.

It is well known that the faces of any embedding traverses every edge of the graph exactly twice. Therefore, the unique face of any 1-face embedding, orientable or not, is a closed walk in the graph which traverses every edge exactly twice. Walks traversing every edge exactly twice are called *double traces* of G. Not every double trace corresponds to 1-face embedding in an orientable surface. However, double traces that traverses each edge once in each of its two directions and induce cyclic permutations at all vertices correspond to 1-face embedding in orientable surfaces and are called *antiparallel strong traces (AST)* (c.f. [4]). The relationship between strong traces and embeddings is summarized by the following theorem, see [4] for more details and further properties of various types of traces.

THEOREM 2.1. *Any antiparallel strong trace S uniquely determines a graph G. Furthermore, there is 1-to-1 correspondence between 1-face embeddings in orientable surfaces of G and antiparallel strong traces S of G.*

Unless stated otherwise, by trace we always mean antiparallel strong trace. A *partial trace* is a walk t in a graph G such that there exists a walk t' in G for which appending t' at the end of t yields a trace in G.

In different contexts it is convenient to represent traces as sequences of either edges or vertices. Since the graphs are simple, both these representations are unambiguous. To stress that we use a particular representation, we use the terms *edge trace* and *vertex trace*, respectively. Finally, as with faces, since traces are closed walks, we implicitly identify the equivalence classes of strong traces corresponding to cyclic shifts of the starting point.

When we append an element to the end of a partial trace t, usually a vertex v, we denote the concatenated (partial) trace by $t \odot v$.

As is customary, we define two embeddings $\Pi_1 = \Pi_1(G)$ and $\Pi_2 = \Pi_2(G)$ to be *isomorphic* if there is an automorphism σ of G such that $(v_1 \dots v_k)$ is a face of Π_1 if and only if $(\sigma(v_1)\dots\sigma(v_k))$ is a face of Π_2. In the case of 1-face embeddings, the embeddings with faces F_1, resp. F_2 are isomorphic if $\sigma(F_1)$ is either F_2 or the reverse of F_2 (recall that we implicitly identify the equivalence class of cyclic shifts of a face).

3 Gap Representation and Canonicalizing Strong Traces.

In this section we introduce three gap representations, vertex, edge, and bio gap. Our main goal is to design linear-time isomorphism test (Theorem 3.4) based on canonical form of vertex gap `CanonGAP` (Theorem 3.2). The edge and bio gap representations are natural variants of vertex gap. Since edge and bio gap are not utilized in the canonical form and isomorphism testing, the reader interested only in the isomorphism algorithm may wish to skip them. On the other hand, the definitions are almost identical so we present them here as well. Furthermore, edge and bio gap seem interesting on their own from both theoretical and practical perspective and since they are currently not very well understood, they might warrant further investigation. In Section 5.2 we present experimental results on using all gap representations for pruning and on empirical data on their distributions in our data sets.

For an illustrative example of the definitions, see Fig. 1 and Tab. 4 in the Appendix. We start with vertex and edge gaps which, for each element t_i of the trace, contain the cyclical distance from t_i to the occurrence of the same element.

DEFINITION 3.1. (VERTEX AND EDGE GAP REPR.) *Let $t = (t_0, t_1, \dots, t_{2m-1})$ be a vertex (resp. edge) strong trace of a graph G. The* vertex, *resp.* edge gap representation $\texttt{GAP}(t)$, *resp.* $\texttt{EGAP}(t)$, *is the vector $(g_0, g_1, \dots, g_{2m-1})$, where g_i is the minimum positive integer such that $t_i = t_j$ and $j = i + g_i$* $(\mathrm{mod}\ 2m)$.

Bio gap representation differs from vertex and edge gaps by taking the minimum of the distances in both directions when calculating the gap for t_i.

DEFINITION 3.2. (BIO GAP REPRESENTATION) *Let $t = (e_0, e_1, \dots, e_{2m-1})$ be an edge strong trace of a graph G. For each i, let a_i and b_i be minimum positive integers such that $e_i = e_{i+a_i \ (\mathrm{mod}\ 2m)}$ and $e_i = e_{i-b_i \ (\mathrm{mod}\ 2m)}$. The* biological gap *$g_i$ is the the minimum of a_i and b_i, and the* biological gap representation *is the vector $\texttt{BioGAP}(t) = (g_0, g_1, \dots, g_{2m-1})$.*

While it would be conceivable to consider also a variant of bio gap for vertex traces, in the self-assembling scenario the binding occurs over edges, not vertices, and thus we do not explore this possibility further.

THEOREM 3.1. *Given a strong trace t corresponding to a 1-face embedding of a graph G, each of $\texttt{GAP}(t)$, $\texttt{EGAP}(W)$, and $\texttt{BioGAP}(t)$ can be computed in time $O(m)$.*

Proof. The reader is referred to Appendix A for details of the proof.

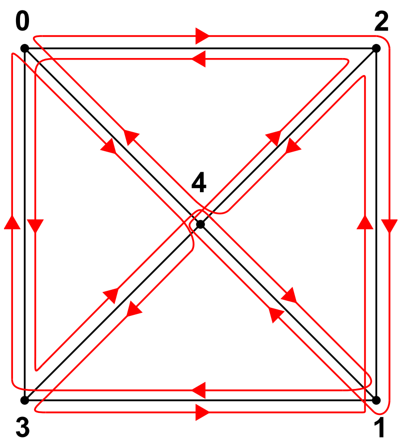


Figure 1: Shown is a square pyramid with vertex strong trace $t = (0,2,1,4,2,0,3,4,1,3,0,4,3,1,2,4)$. Here, $\texttt{GAP}(t) = (5,3,6,4,10,5,3,4,5,3,6,4,10,5,3,4)$. The gap representation of the reversed walk t^{-1} is $\texttt{GAP}(t^{-1}) = (4,10,5,3,4,5,3,6,4,10,5,3,4,5,3,6)$. Note, $\texttt{GAP}(t^{-1})$ is not the reversal of $\texttt{GAP}(t)$. The canonical representation is then obtained by taking the lexicographic minimum of $\texttt{GAP}(t)$ and $\texttt{GAP}(t^{-1})$. Thus, we get $\texttt{CanonGAP}(t) = (3,4,5,3,6,4,10,5,3,4,5,3,6,4,10,5)$ which corresponds to a cyclic shift of $\texttt{GAP}(t^{-1})$. The edge strong trace (an edge $\{i,j\}$ is just written as ij or ji) is $t_E = (02,21,14,42,20,03,34,41,13,30,04,43,31,12,24,40)$. We obtain $\texttt{GAP}(t_E) = (4,12,5,11,12,4,5,11,4,12,5,11,12,4,5,11)$ and $\texttt{CanonGAP}(t_E) = (4,5,11,4,12,5,11,12,4,5,11,4,12,5,11,12)$. The biological gap is $\texttt{BioGAP}(t_E) = (4,4,5,5,4,4,5,5,4,4,5,5,4,4,5,5)$. Here t is the only strong trace for the square pyramid that has smallest max bio gap 5, cf. Tabular 4, where this specific embedding has ID 4.

For a vector $W = w_1 w_2 \dots w_k$, let W^R denote the reverse $w_k \dots w_2 w_1$ of W.

Definition 3.3. *Let t be a strong trace on the graph G, then the* canonical representation *of t, denoted by $\mathtt{CanonGAP}(t)$, is defined by*

$$\mathtt{CanonGAP}(t) = \text{lex\_min}\{\text{lex\_min}(\texttt{GAP}(t)), \text{lex\_min}(\texttt{GAP}(t^R))\},$$

where lex_min *represents a function that returns the lexicographical smallest shift of a given vector.*

Theorem 3.2. *Let t_1 and t_2 be strong traces corresponding to 1-face embeddings $\Pi_1 = \Pi_1(G)$ and $\Pi_2 = \Pi_2(G)$. Then $\mathtt{CanonGAP}(t_1) = \mathtt{CanonGAP}(t_2)$ if and only if Π_1 and Π_2 are isomorphic.*

Proof. We give here a sketch of the proof; for full details see the Appendix. Our proof is based on several observations. First, if a vector W is the vertex gap of an AST t, then W determines t uniquely up to isomorphism; we call the embedding corresponding to t the *associated embedding* of W. Second, while $\texttt{GAP}(t)$ depends on the starting point of t, the cyclic shifts of $\texttt{GAP}(t)$ are in 1-to-1 correspondence with cyclic shifts of t. In particular, there is an unique embedding associated with all shifts of $\texttt{GAP}(t)$. The final ingredient is that choosing the minimum among shifts of gaps of t and t^R leads exactly to identification of Π and Π^R in our enumeration.

Theorem 3.3. *Given a strong trace t corresponding to a orientable 1-face embedding of a graph G, $\mathtt{CanonGAP}(t)$ can be computed in time $O(m)$.*

Proof. By Theorem 3.1, it is possible to compute the $\texttt{GAP}(t)$ and $\texttt{GAP}(t^R)$ in linear time. Additionally it was shown in [34] that finding the lexicographically smallest shift of a vector can be done in $O(m)$. It follows directly that $\texttt{CanonGAP}(t)$ can be computed in $O(m)$.

Theorem 3.3 immediately implies that there is a linear-time algorithm deciding isomorphism of two embeddings whose running time does not depend on the size of the automorphism group of the graph.

Theorem 3.4. *There is an algorithm deciding isomorphism of any two 1-face embeddings in time $O(m)$.*

4 Algorithms.

The basis for all used algorithms is a branching strategy for generating strong traces, as given in Algorithm 1. This general strategy depends on the following three routines, whose specifications leads to a particular algorithm.

isFeasible(t,v) returns true if appending v to t would lead to a valid partial strong trace. This routine is the same in all the used variants of the algorithm.

isCanonical(t,v) returns true if appending v to t might lead to a canonical strong trace.

isCanonical(t) returns true if t is the canonical representative of its equivalence class.

We now describe two specifications of $\texttt{isCanonical}(t,v)$ and $\texttt{isCanonical}(t)$, giving our version of the AEST algorithm from [4] and our algorithm GapEST based on canonical gap representations, respectively. We will use $t \prec t'$ to denote that a trace t is lexicographically smaller than t'.

We start with the description of $\texttt{isCanonical}(t,v)$ in the AEST algorithm. The algorithm assumes that $v_0 v_1$ is always an edge of G and that all strong traces start with this edge. From each set of pairwise isomorphic strong traces, the algorithm chooses the canonical strong trace defined as the strong trace that is lexicographically minimum. To this end, the algorithm keeps a set S_A of automorphisms of the graph, which is used to cut branches of the search tree that cannot lead to a lexicographically minimum strong trace. At each step, all automorphisms in S_A fix the current partial trace t.

Algorithm 1 Recursively extend a partial strong trace

```
Require: $t$ is a partial strong trace
1: function ExtendTrace($t$)
2:     if $|t| < 2m$ then                          ▷ $t$ is not a complete trace
3:         $u \leftarrow$ the last vertex of $t$
4:         for all $v \in N(u)$ do
5:             if isFeasible($t, v$) and isCanonical($t, v$) then
6:                 Update()
7:                 ExtendTrace($t \odot v$)
8:     else               ▷ $|t| = 2m$ and $t$ is a complete strong trace
9:         if isCanonical($t$) then
10:            Add $t$ to the list of solutions
```

The set S_A is then used to determine which traces can lead to a canonical trace in three rounds. Each of these phases takes during the verification whether a partial strong trace can be extended by a vertex v. In the first round, when t is being extended, S_A is used to cut the branching extending t by vertices which are equivalent under some automorphism in S_A by choosing a minimal representative v among these vertices. Furthermore, all the elements of S_A that do not fix v are removed from S_A at this step. The description of the remaining two rounds is based solely on our own implementation of AEST algorithm; the original paper [4] does not provide sufficient details to fully reconstruct the approach. We divide S_A into three independently maintained subsets S_A, S_A^S, and S_A^{SR} used in the respective phases. Initially, all three sets equal $\mathrm{Aut}(G)$. From a high-level point of view, these sets represent automorphisms which can lead to a canonical trace in different ways. The automorphisms in S_A are candidates for producing a canonical trace directly. For the remaining two sets, after applying an automorphism a canonical trace could be obtained after a shift (S_A^S), resp. after a shift and reversal (S_A^{SR}). Therefore, the set S_A takes the role of S_A as described above and is used and maintained in the same way. For each automorphism σ in $S_A^S \cup S_A^{SR}$, a trace r is produced to verify whether the current partial trace t' can be canonical. If σ is in S_A^S, then $r = \sigma(t')$ and if σ is in S_A^{SR}, then r is the reverse of $\sigma(t')$. Since each canonical trace starts with v_0v_1, the split into S_A^S and S_A^{SR} implies that it is sufficient to focus on the case when r contain v_0v_1. Indeed, if v_0v_1 is not in r, then we do not know how much the trace will be shifted and thus we cannot decide whether t' can be canonical. Consequently, if an automorphism in S_A^S or S_A^{SR} does not map any edge to v_0v_1 (resp. v_1v_0), then the current branch cannot be pruned and we cannot remove any automorphism from S_A^S, resp. S_A^{SR}. On the other hand, if v_0v_1 is contained in r, then the shift is given uniquely. Once the shift is determined, the algorithm compares the current partial trace t' with s, the unique shift (resp. shift and reversal) as follows. If $s \prec t'$, then t' cannot be canonical (since t' is equivalent to $\sigma^{-1}(s)$ and $s \prec t'$). Therefore, if $s \prec t'$, then this branch of the search tree can be pruned. On the other hand, if $t' \prec s$, then s cannot be canonical. Since S_A^S and S_A^{SR} should contain only candidate automorphisms for producing a canonical trace, in this case we can remove σ from S_A^S, resp. from S_A^{SR}. If we cannot assert neither $s \prec t'$ nor $t' \prec s$, then we proceed by further extending t' and not changing S_A^S and S_A^{SR}. If a partial trace t' is not pruned from the search tree, this process continues until a full-length trace t with $|t| = 2m$ is produced. The fact whether the full-length trace t is canonical is decided by isCanonical(t) using the unique shifted trace s and the same rules that are used to decide whether the partial traces can be canonical.

To design a basic enumeration algorithm GapEST based on Algorithm 1, we can set isCanonical(t, v) to always return true, and isCanonical(t) to return true if CanonGAP(t) is different from CanonGAP(t') of all previously found solutions t'. (We use a hash table to store the gaps.) However, in practice it is beneficial to include a pruning strategy similar to S_A, as described above, into isCanonical(t, v), but it is not necessary for S_A at the start to contain the all automorphisms of G. Indeed, when the sole number of automorphisms is an obstacle in computing the non-isomorphic embeddings, S_A can still contain an arbitrary subset of automorphisms and be used for pruning. This is illustrated in the next section on trees, where S_A equal to a generating set of $\mathrm{Aut}(G)$ is used.

Note that a naïve exhaustive search version of Algorithm 1 could be obtained by using an isCanonical(t, v) function that always returns true and isCanonical(t) which compares t to all solutions found so far using every automorphism σ of G, comparing all shifts of $\sigma(t)$ and all shifts of the reversal of $\sigma(t)$ to all previous solutions.

To implement the pruning, the condition on Line 5 of Algorithm 1 is extended to contain also a verification isOptimal(t, v), which returns true if $t \odot v$ does not violate the required condition on gaps. For vertex gap, clearly the gap vector can be computed online, that is, in such a way that only the gap for v and the last vertex in t needs to be computed. Such an incremental computation of the gap vector can be done in constant time for all variants of gaps.

5 Experiments.

The source code was written in C++ and compiled with gcc ver. 5.2.1 (Ubuntu 5.2.1-22ubuntu2). External libraries Boost Graph Library [21] and the bliss library [23] were used for basic manipulation of graphs and identification of the generators of the automorphism group of graphs.

All computations were performed on a cluster consisting of 20 Dell c8220 servers each with two 2.8Ghz Intel Ivybridge 10-core CPUs (E5-2680v2) (i.e. 400 cores), each pair of CPUs shared at least 128 GB Ram and a 200 GB SSD harddisk. Alls CPU were used exclusively in order to minimize side-effect when testing computation times.

5.1 Data Sets. We evaluate the performance of our algorithms on all simple graphs with at most 9 vertices, on all cubic graphs with at most 22 vertices, and all trees with 15 – 20 vertices. Most of these data sets contain from thousands to hundreds of thousands graphs, see below.

By Euler formula Eq. (2.1), only graphs with $|E|-|V|$ odd might have a 1-face embedding; such graphs are called *feasible instances*. In particular, a cubic graph is feasible if and only if it has $2 \pmod 4$ vertices. We denote the class of all feasible connected with graphs 9 vertices by $\mathcal{G}_9$, trees on i vertices by $\mathcal{T}_i$, and all cubic graphs on 18 and 22 vertices by $\mathcal{C}_{18}$ and $\mathcal{C}_{22}$, respectively (the trees and the latter two classes are all feasible).

The set $\mathcal{G}_9$ contain 130 553 graphs; their distribution according to connectivity is stated in Table 3. There are in total 41 301, resp. 7 319 447 graphs in $\mathcal{C}_{18}$ and $\mathcal{C}_{22}$. In $\mathcal{C}_{18}$, graphs having connectivity 1–3 constitute 3.5, 22.8, and 73.8 percent of the total, while in $\mathcal{C}_{22}$ it is 1.8, 17.5, and 80.7 percent, respectively. (No cubic graph can have connectivity 4 or more.) In both of these sets, graphs with girth 3–6 constitute approximately 81, 18, 1, resp. less than 0.1 percent of the total. There are in total less than 1000 feasible cubic graphs on at most 14 vertices and in total less than 1000 connected graphs on at most 7 vertices, we omit the analysis for these small sets. All graphs were obtained from `https://hog.grinvin.org/`.

We have chosen the set of all connected graphs as a baseline. Cubic graphs and trees were chosen due to their suitable size and because both sets have interesting properties from a topological graph theory perspective. As any other graph, every tree has $\Pi_v(\deg_v - 1)!$ distinct combinatorial embeddings. However, all these embeddings have one face and are in the sphere. Thus enumerating all 1-face embeddings of trees, regardless of automorphisms, is trivial. Cubic graphs are sparse and have a relatively small number of embeddings and automorphisms for their size. Large subclasses of cubic graphs are also central to the intersection of colorings and topology through results such as the Four-color Theorem or (the disproving of) the Grunbaum conjecture [26], and thus their embeddings are intensively studied, see for example [5, 17, 27, 29].

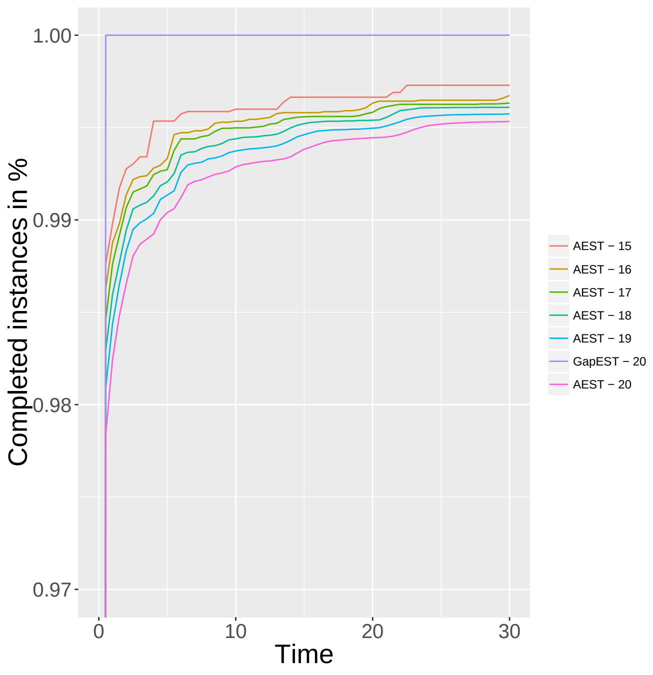
Even for general graphs as small as seven vertices and sparse graphs with around 10 vertices, enumerating topological invariants might require running time in the order of magnitude in hundreds of hours, see [6, 9, 14, 33]. Therefore, our data sets are both challenging and large enough to evaluate the efficiency of the algorithms. Furthermore, the finer partitionings of the data sets according to the number of edges, the size of the automorphism group, or the connectivity, yield data sets which still contain sufficiently-many graphs and are difficult enough to gain meaningful insights into the performance of the algorithms. We consider our choices of parameters to further partition the data sets natural, and in particular we focus on girth and connectivity due to their known relationships with the maximum genus [7, 8, 20, 38].

5.2 Results. We start with the performance of `AEST` and `GapEST` on trees. The method `GapEST` was able to compute all 1-face embedding for any tree with at most 20 vertices in less than half a second. In contrast, there are instances where `AEST` was not able to compute all 1-face embeddings within a time-limit of 30 seconds, see Fig. 2. In particular, `AEST` could not solve 2 273 of the trees with 20 vertices. Among the instances intractable in 30 seconds for `AEST` are graphs with stars $K_{1,n}$ as subgraphs for large n (see Fig. 8d, Appendix). The latter indicates that the GapEST algorithm is more suitable for graphs with a large number of automorphisms.

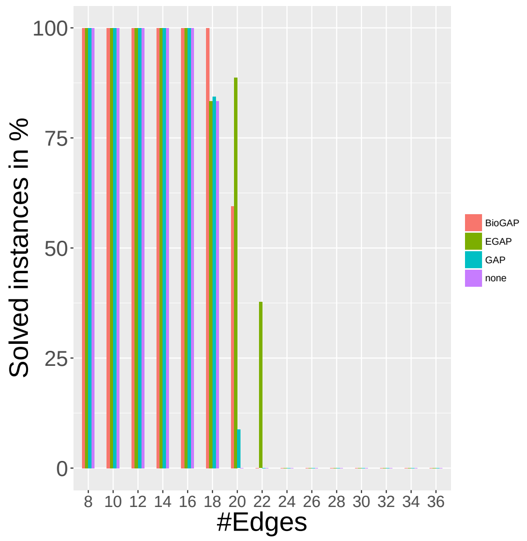
On the other hand, if the number of embeddings increases, the performance of `GapEST` degrades due to memory issues, since it needs to store each solution. This, in particular, makes running-time comparisons difficult, see Section 5.4 for further discussion of the issue. For the remaining data sets, a significant fraction of the graphs has a very large number of embeddings (see Fig. 3b). Moreover, the number of automorphisms of the graphs in the remainining considered classes is relatively modest and `AEST` outperforms `GapEST` on our hardware. Therefore, in the following we focus on the enumeration properties of the datasets and report results only for `AEST`.

For 56,4% of the graphs in $\mathcal{G}_9$ all pairwise non-isomorphic 1-face embeddings could be computed within a time-limit of 30 seconds with our implementation of `AEST`, see Fig. 3a for the running times. For all the instances in $\mathcal{G}_9$ that were not solved, we stored the number of pairwise non-isomorphic 1-face embeddings computed in 30 seconds and the number of computed 1-face embeddings for these instances is always larger than 10^6, see Fig. 3b. Fig. 4a shows the number of 1-face embeddings as a function of all combinatorial embeddings of the solved $\mathcal{G}_9$ instances. The number of pairwise non-isomorphic 1-face embeddings scales almost linearly with the number of all embeddings for a fixed $|\mathrm{Aut}(G)|$, with the slope depending on $|\mathrm{Aut}(G)|$. The latter is also reflected in Fig. 4b that shows the percentage of 1-face embeddings as a function of the number of rotation systems for all solved $\mathcal{G}_9$ instances that have more than 1000 1-face embeddings

In the set $\mathcal{G}_9$, all instances with at most 16 edges and 83.4% of the instances with 18 edges can be solved in 30 seconds, while none of the instances with at least 20 edges can be solved. The violin plot in Fig. 5a indicates that the number of 1-face embedding increases exponentially with the number of edges for instances in $\mathcal{G}_9$ with 8–18 edges. The mean number of 1-face embeddings among the



(a)



(b)

Figure 2: (a) The relative number of completed instances for the classes of trees on 15 up to 20 vertices as a function of the runtime. `AEST`-i denotes `AEST` on $\mathcal{T}_i$ and `GapEST`-20 refers to the method `GapEST` applied to $\mathcal{T}_{20}$. For simplicity, we only present runtimes for `GapEST`-20, as the runtime on trees with less than 20 vertices is smaller. (b) Different pruning strategies for $\mathcal{G}_9$ instances. "None" means that no pruning strategy is applied.

solved graphs is indicated in the underlying boxplot. The violin plot in Fig. 5b shows the number of pairwise non-isomorphic 1-face embeddings as a function of the edge-connectivity for the solved $\mathcal{G}_9$ instances. The mean increases from connectivity 1 to 3, and decreases from 3 to 4, and there is a notable change of shape of the distributions

For each graph in $\mathcal{C}_{22}$, our implementation of `AEST` is able to find all pairwise non-isomorphic embeddings in at most 4.5 seconds, with the average being 1.88 seconds per instance, with the total runtime on $\mathcal{C}_{22}$ corresponding to 159.14 days of single-core computations. Out of the $\mathcal{C}_{22}$ instances, 7 219 968 (98.64%) have at least one 1-face embedding. The number of 1-face embeddings ranges between 38 and 464 896 and 80.2% of these instances have at least 10^5 1-face embeddings. The number of 1-face embeddings as a function of connectivity for $\mathcal{C}_{22}$ instances is shown in Fig. 5d, the number of embeddings increases with the increasing connectivity. The violin plot in Fig. 5c shows the number of 1-face embeddings as a function of the girth that varies between 3 and 6 for the $\mathcal{C}_{22}$ instances. Among the graphs that have at least one 1-face embedding, for girth 5 no instance has less than 1640 1-face embeddings and for girth 6 no instance has less than 19 456 1-face embeddings. Fig. 6a depicts the runtime as a function of the number of all combinatorial embeddings for all the instances in $\mathcal{C}_{22}$ with at least one 1-face embedding.

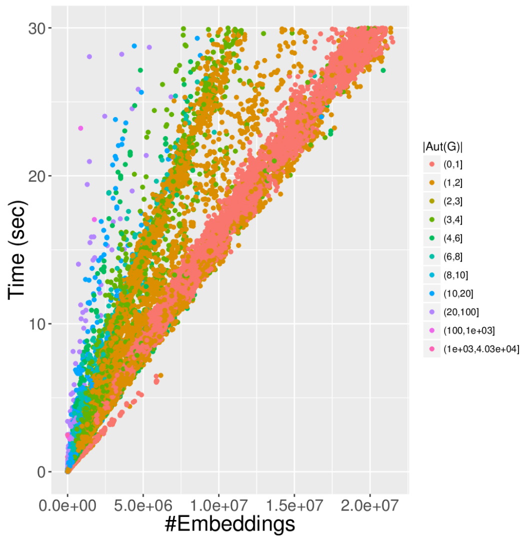
The results of applying different pruning strategies for the $\mathcal{G}_9$ instances are in Fig. 2b. To be more precise, let $f \in \{\texttt{GAP}, \texttt{EGAP}, \texttt{BioGAP}\}$ and for a given trace t let $\max(f(t))$ be the largest value in $f(t)$. A trace t of a graph G is *f-optimal* if $\max(f(t))$ is the smallest among all traces of G. To prune the search space, we discard any further computation for partially computed (vertex, edge, bio) gap representation whenever the largest gap within this partial gap representation is already larger than the f-optimal strong trace w.r.t. $\mathcal{T}$, where $f \in \{\texttt{GAP}, \texttt{EGAP}, \texttt{BioGAP}\}$ and $\mathcal{T}$ is the set of 1-face embeddings found so-far. With our 30 second time limit, the baseline version of `AEST` solves 83.4% of all the $\mathcal{G}_9$ instances with 18 edges and the version pruning w.r.t. the `BioGAP` solves all instances with 18 edges. For instances with more than 24 edges none of the pruning strategies could find all f-optimal 1-face embeddings within the time-limit of 30 seconds for any of the instances. Altogether, pruning w.r.t. `EGAP` leads to the highest speedup on $\mathcal{G}_9$.

In Fig. 6b (resp. Fig. 6c) we give the results w.r.t. optimization of the vertex gap (resp. bio gap). The number of embeddings depicted refers to the number of f-optimal embeddings found w.r.t. corresponding optimization criteria. The average runtime when optimizing the vertex gap (resp. bio gap) was 0.23sec (resp. 0.52sec). Note that many

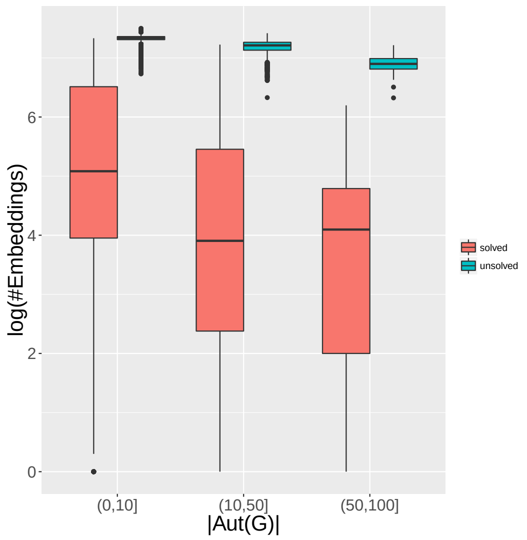
Connectivity:	1	2	3	4	≥ 5
Graphs in $\mathcal{G}_9$:	63 638	113 256	68 715	14 306	1 160

$\sharp$ vertices n:	15	16	17	18	19	20
Trees on n vertices:	7 741	19 320	48 629	123 867	317 955	823 065

Table 3: Number of graphs in subsets of data sets.

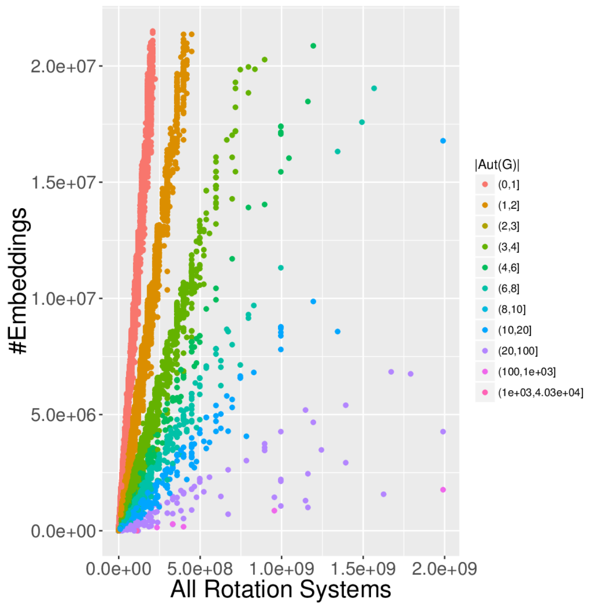


(a)

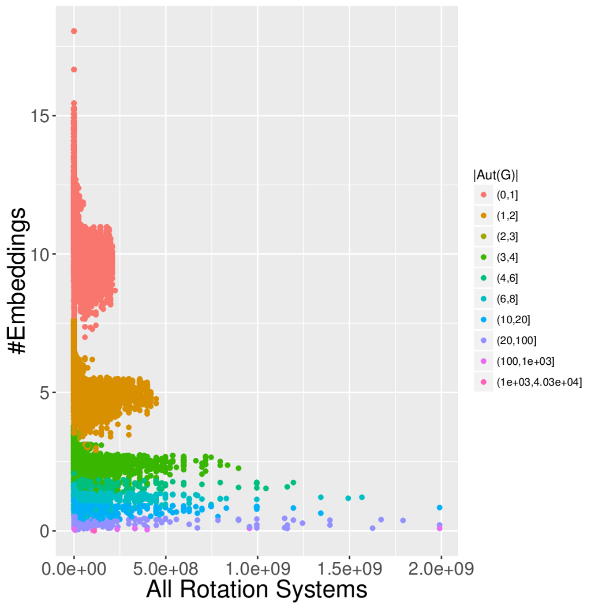


(b)

Figure 3: (a) The runtime as a function of the number all combinatorial embeddings depending on $|\text{Aut}(G)|$ in $\mathcal{G}_9$; (b) the number of pairwise non-isomorphic 1-face embeddings found in 30 seconds in solved and unsolved $\mathcal{G}_9$instances, grouped by $|\text{Aut}(G)|$.

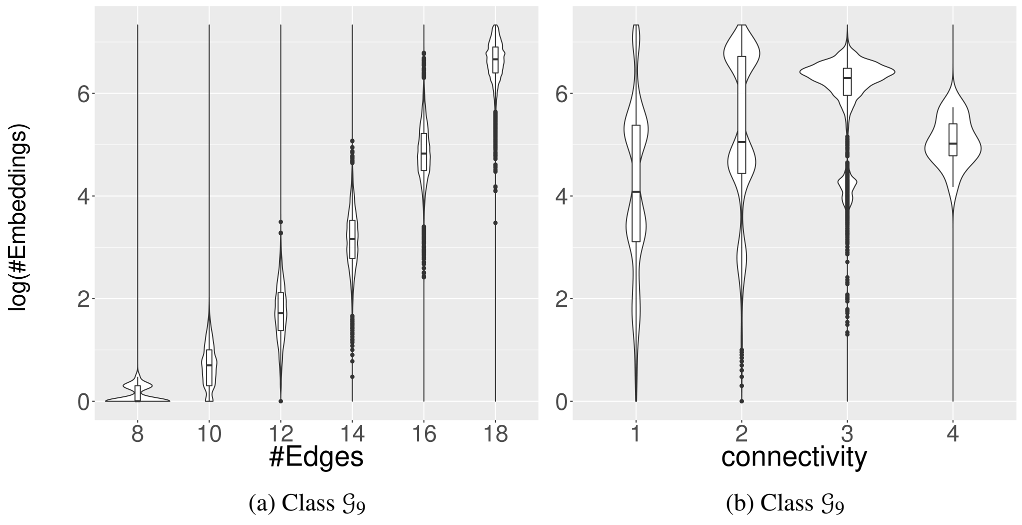


(a) Class $\mathcal{G}_9$

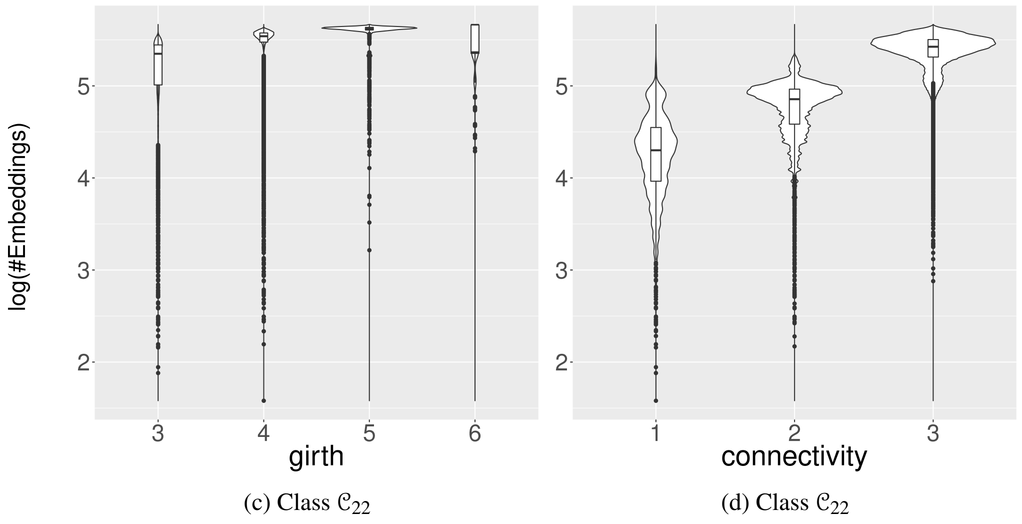


(b) Class $\mathcal{G}_9$

Figure 4: (a) The number of pairwise non-isomorphic 1-face embeddings as a function of all combinatorial embeddings; (b) The number of pairwise non-isomorphic 1-face embeddings as a percentage of all combinatorial embeddings.

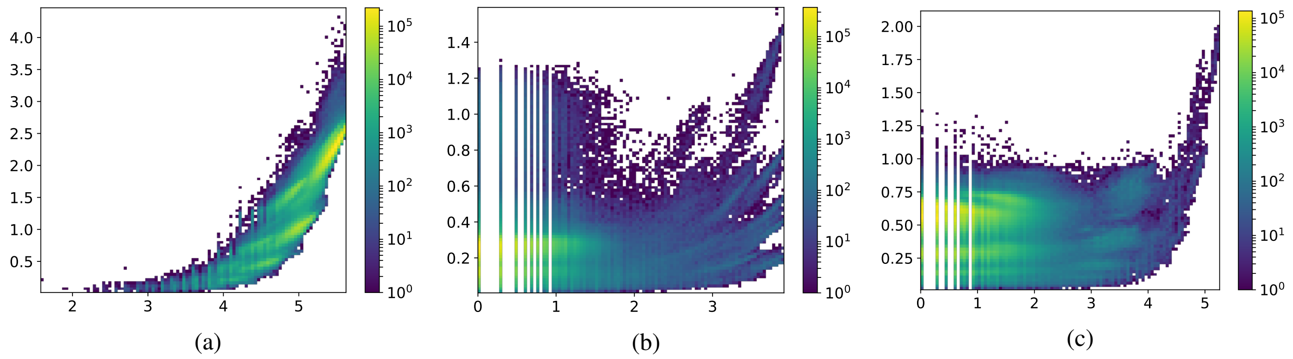


(a) Class $\mathcal{G}_9$ (b) Class $\mathcal{G}_9$



(c) Class $\mathcal{C}_{22}$ (d) Class $\mathcal{C}_{22}$

Figure 5: Number of 1-face embeddings ($\log_{10}$-scaled) as a function of the number of edges and the edge-connectivity for the solved $\mathcal{G}_9$ instances (a), (b) and as function of the edge-connectivity and girth for the class $\mathcal{C}_{22}$ (c), (d) are shown. Note, each $\mathcal{C}_{22}$ instance has 33 edges and thus, instead of drawing the plot w.r.t. the number of edges we used the girth.



(a) (b) (c)

Figure 6: Heatmap plots for the class $\mathcal{C}_{22}$: depicted are runtimes in seconds (vertical axis) and number of all possible embeddings (horizontal axis) for all solved instances (left), runtimes and number of all possible f-optimal embeddings for all solved $\mathcal{C}_{22}$ instances with $f =$ `GAP` (middle) and $f =$ `BioGAP` (right); the number of (optimal) embeddings is given in $\log_{10}$ scale.

of the instances have only very few optimal embeddings, i.e. 3 581 198 graphs (49.6%) have 4 or less optimal 1-face embeddings w.r.t. the vertex gap, and 5 518 929 graphs (76.4%) have 4 or less optimal 1-face embeddings w.r.t. the bio gap. The results for optimization w.r.t. the edge gap are not depicted, but they are very similar to the result for computing all embeddings, as most of the instances have girth equal to 3 and it can be shown that $\max(\mathtt{EGAP}(t)) = 3$ for any trace t of a cubic graph with girth 3.

5.3 Implications. The main consequences of our experiments are the following. First, there is a very fast linear-time algorithm to decide whether two 1-face embeddings are isomorphic. Considering how time-consuming are the computations in topological graph theory, our experiments suggest that the studied enumeration algorithms are fast and scale well. Our results enable one to choose whether to use AEST or GapEST to compute the number of pairwise non-isomorphic 1-face embeddings based on the number of automorphisms of the graphs in the class of interest, and to understand the limitations of the approaches when applied on scale.

5.3.1 Ratio of 1-face embeddings and pairwise non-isomorphic 1-face embeddings to all combinatorial embeddings. As stated before, these ratios were not known before, even up to an order of magnitude, in nontrivial datasets. Currently, there is no evidence suggesting even that 1-face embeddings constitute a significant proportion of all embeddings. Our results show that as many as 10% of *all* embeddings of graphs in $\mathcal{C}_{22}$ without a nontrivial automorphism are pairwise non-isomorphic 1-face embeddings (see Fig. 4b). Furthermore, there seems to be a linear dependence of the number of non-isomorphic 1-face embeddings on the number of automorphisms and the total number of embeddings. Consequently, a use of this dependence might provide a good estimate of the number of pairwise non-isomorphic 1-face embeddings of the graph. Since the number of pairwise non-isomorphic embeddings is a rather large constant fraction of all embeddings, random sampling in the space of all embedding in conjunction with our linear-time isomorphism test might yield a viable method of obtaining very large sets of pairwise non-isomorphic 1-face embeddings. A somewhat surprising observations are the decrease of the number of pairwise non-isomorphic 1-face embeddings in 4-connected graphs and the multimodal shape of the density of the number of pairwise non-isomorphic 1-face embeddings for connectivity 1 and 2. At present, we have no explanation for these phenomena.

5.3.2 Gap vectors. Although the gap vectors seem very natural and interesting on their own, to the best of our knowledge they were not studied before. Our results show that the number of optimal 1-face embeddings is very small for the majority of graphs, which leads to a significant speedup in the enumeration of 1-face embeddings. Both these results further indicate that gap vectors deserve additional investigation.

5.4 Limitations. The main drawback of our work is that we are able to compare the running times of AEST and GapEST on large datasets only to a limited extent. Roughly speaking, this is caused by the fact that the two algorithms use exponential amount of resources in a different way. On one hand, AEST needs to compute explicitly all elements (not just the generators) of the full automorphism group to find even a *single pair* of non-isomorphic 1-face embeddings, and this computation rapidly gets lengthy. On the other hand, to find the number of *all* pairwise non-isomorphic 1-face embeddings, GapEST needs to store all solutions in the memory. Since each embedding is represented by roughly $2|E(G)|$ integers, and the number of solutions easily grows to over 10^7 even in our datasets, the physical cores quickly start to compete for the shared memory. This skews the measurements and, since even one difficult instance can influence the running times on all the other cores with shared memory, makes the measurements unpredictable. Consequently, this problem limits the extent to which we can use parallelization in the experiments and thus, due to the sizes of the data sets, also our ability to perform comparisons at such a large scale. On the other hand, restricting the size of the data sets would significantly limit the insight gained from the results.

6 Conclusion.

In this work we have introduced three novel invariants called *gap representations*. We proved that the canonical form of the vertex gap representation is an isomorphism invariant of 1-face embeddings. We provided a linear-time algorithm to compute the canonical form and consequently, we have obtained a new method GapEST for enumerating pairwise non-isomorphic 1-face embeddings of an arbitrary graph.

To evaluate our GapEST algorithm and a previous approach AEST, we have performed first large-scale computational survey of embeddings. Our experiments indicate that GapEST algorithm is more suitable for graphs with large number of automorphisms, while AEST is more suitable for graphs with large number of pairwise non-isomorphic 1-face embeddings. In particular, GapEST algorithm clearly outperforms AEST on trees. Since the computation of all automorphisms is a bottleneck for AEST, GapEST is also more suitable when either a large set of 1-face embeddings (as opposed to their full census), or lower bounds on their number are sought. Further advantage of GapEST is that it is easily modified to take into account gap (distance) properties of embeddings to achieve speed-up by pruning, or when only a subset of 1-face embeddings is sought.

The obtained data sets provide the first insight into the behavior of embeddings on large scale. Among the most interesting results are the large portion of all combinatorial embeddings being pairwise non-isomorphic 1-face embeddings, and, for fixed $|\mathrm{AUT}(G)|$, the seemingly linear dependence between the number of 1-face embeddings number and the number of all embeddings. For cubic graphs, our results show that (i) almost all (98.64%) of 22-vertex graphs are upper-embeddable, (ii) the lower bound on the number 1-face embeddings significantly increases with the girth; and (iii) majority of the graphs have at least 10^5 pairwise non-isomorphic embeddings. In all these cases, the order of the magnitude of the values of the parameters was previously not known.

Acknowledgements. This contribution is supported in part by the Independent Research Fund Denmark, Natural Sciences, grant DFF-7014-00041.

References

[1] `https://github.com/nojgaard/DoubleTrace`.

[2] `http://www.imada.sdu.dk/Employees/daniel/gapEST/`.

[3] D. Archdeacon, L. Goddyn, and J. Rus. E-restricted double traces. E-print ArXiv:1610.09888, 2016.

[4] N. Bašić, D. Bokal, T. Boothby, and J. Rus. An algebraic approach to enumerating non-equivalent double traces in graphs. *MATCH Commun. Math. Comput. Chem.*, 78(3):581–594, 2017.

[5] s-m. Belcastro and J. Kaminski. Families of dot-product snarks on orientable surfaces of low genus. *Graphs Combin.*, 23(3):229–240, 2007.

[6] S. Beyer, M. Chimani, I. Hedtke, and M. Kotrbčík. A practical method for the minimum genus of a graph: Models and experiments. In A. V. Goldberg and A. S. Kulikov, editors, *Proceedings of 15th International Symposium on Experimental Algorithms SEA'16*, pages 75–88. Springer International Publishing, 2016.

[7] J. Chen, D. Archdeacon, and J. L. Gross. Maximum genus and connectivity. *Discrete Math.*, 149(1):19 – 29, 1996.

[8] J. Chen, S. P. Kanchi, and J. L. Gross. A tight lower bound on the maximum genus of a simplicial graph. *Discrete Math.*, 156(1):83 – 102, 1996.

[9] M. Conder and R. Grande. On embeddings of circulant graphs. *Electronic J. Combin.*, 22(2):♯ P2.28, 2015.

[10] I. Drobnak, A. Ljubetič, H. Gradišar, T. Pisanski, and R. Jerala. Designed protein origami. In Aitziber L. Cortajarena and Tijana Z. Grove, editors, *Protein-based Engineered Nanostructures*, pages 7–27, Cham, 2016. Springer International Publishing.

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

[12] H. Gradišar, S. Božič, T. Doles, D. Vengust, I. Hafner-Bratkovič, A. Mertelj, B. Webb, A. Šali, S. Klavžar, and R. Jerala. Design of a single-chain polypeptide tetrahedron assembled from coiled-coil segments. *Nature chemical biology*, 9(6):362–366, 2013.

[13] J. L. Gross. Embeddings of graphs of fixed treewidth and bounded degree. *Ars Mathematica Contemporanea*, 7:379–403, 2014.

[14] J. L. Gross, I. F. Khan, T. Mansour, and T. W. Tucker. Calculating genus polynomials via string operations and matrices. Manuscript, 2016, http://www.cs.columbia.edu/~gross/research/StringOps-GKMT-2016.pdf.

[15] J. L. Gross, T. Mansour, T. W. Tucker, and D. G-L. Wang. Log-concavity of combinations of sequences and applications to genus distributions. *SIAM J. Discrete Math.*, 29:1002–1029, 2015.

[16] J. L. Gross and Furst M.L. Hierarchy for imbedding-distribution invariants of a graph. *J. Graph Theory*, 11(2):205–220, 1987.

[17] J. L. Gross and T Sun. Genus distribution of 3-regular series-parallel graphs. *Discrete Math. Theor. Comput. Sci.*, 16(3):129–146, 2014.

[18] J.L. Gross, D.P. Robbins, and T.W. Tucker. Genus distributions for bouquets of circles. *J. Combin. Theory Ser. B*, 47(3):292 – 306, 1989.

[19] J.L. Gross and W.T. Tucker. *Topological Graph Theory (Dover Books on Mathematics)*. John Wiley & Sons, New York, US, 2001.

[20] Y. Huang. Maximum genus and girth of graphs. *Discrete Mathematics*, 194(1):253 – 259, 1999.

[21] A. Lumsdaine J.G. Siek, L.Q. Lee. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[22] M. Jungerman. A charactrerization of upper embeddable graphs. *Trans. Amer. Math. Soc.*, 241:401–406, 1978.

[23] T. Junttila and P. Kaski. Engineering an efficient canonical labeling tool for large and sparse graphs. In D. Applegate, G. S. Brodal, D. Panario, and R. Sedgewick, editors, *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments ALENEX'07*, pages 135–149. SIAM, 2007.

[24] S. Klavžzar and J. Rus. Stable traces as a model for self-assembly of polypeptide nanoscale polyhedrons. *MATCH Commun. Math. Comput. Chem.*, 70:317–330, 2013.

[25] V. Kočar, J. S. Schreck, S. Čeru, H. Gradišar, N. Bašić, T. Pisanski, J. P. K. Doye, and R. Jerala. Design principles for rapid folding of knotted DNA nanostructures. *Nature communications*, 7, 2016.

[26] M. Kochol. Polyhedral embeddings of snarks in orientable surfaces. *Proc. Amer. Math. Soc.*, 137:1613–1619, 2009.

[27] M. Kotrbčík, N. Matsumoto, B. Mohar, A. Nakamoto, K. Noguchi, K. Ozeki, and A. Vodopivec. Grnbaum colorings of even triangulations on surfaces. *J. Graph Theory*, page to appear.

[28] B. Mohar and C. Thomassen. *Graphs on Surfaces*. The Johns Hopkins University Press, 2001.

[29] B. Mohar and A. Vodopivec. On polyhedral embeddings of cubic graphs. *Combin. Probab. Comput.*, 15(6):877–893, 2006.

[30] H. Ren and Y. Bai. Exponentially many maximum genus embeddings and genus embeddings for complete graphs. *Sci. in China Ser. A: Mathematics*, 51(11):2013–2019, 2008.

[31] G. Ringel. *Map Color Theorem*. Springer-Verlag, 1974.

[32] J. Rus. Antiparallel d-stable traces and a stronger version of ore problem. *J. Math. Biology*, 75(1):109–127, 2017.

[33] Peter Schmidt. *Algoritmické vlastnosti vnorení grafov do plôch*. B.Sc. thesis, Comenius University, 2012. in Slovak.

[34] Y. Shiloach. Fast canonization of circular strings. *Journal of Algorithms*, 2(2):107 – 121, 1981.

[35] S. Stahl. Permutation-partition pairs: a combinatorial generalization of graph embeddings. *Trans. Amer. Math. Soc.*, 259:129 – 145, 1980.

[36] S. Stahl. Permutation-partition pairs. ii. bounds on the genus of the amalgamation of graphs. *Trans. Amer. Math. Soc.*, 271:175 – 182, 1982.

[37] C. Thomassen. The graph genus problem is NP-complete. *Journal of Algorithms*, 10:568–576, 1989.

[38] M. Škoviera. The maximum genus of graphs of diameter two. *Discrete Math.*, 87(2):175 – 180, 1991.

Appendix

In Section A we give proofs and in Section B additional figures and results are given (please refer to the main article text for details).

A Proofs.

Proof. [Proof of Theorem 3.1] The gap vectors $g = (g_0, \dots, g_{2m-1})$ can be computed in linear time for example by the following two-pass algorithm. For simplicity, we give the algorithm only for the vertex gap, the other cases are similar. First, initialize a set of empty lists L_v, one for each vertex of G. In the first pass over t, starting with position 1 add to the list L_v all positions of v in t in the order as encountered in the pass over t. Finally, for each L_v, put the last item of L_v also on the position 0. In the second step, pass simultaneously over t, the resulting vector g, and lists L_v, starting at the beginning of all of these vectors. Suppose a vertex v is is encountered at the position i of t, and this is j-th occurrence of v in t. If j equals 1, set g_i to $L_v[0] + L_v[1] \pmod{2m}$, otherwise set g_i to $L_v[j] - L_v[j-1]$.

To prove that CanonGAP is an isomorphism invariant of connected graphs, we first observe the following property of vertex gaps.

Lemma A.1. *Let Π^1 and Π^2 be 1-face embeddings of a graph G and let $W^i = w_1^i w_2^i \dots w_{2m}^i$ be their vertex gap vectors. If $w_i^1 = w_i^2$ for all i, then Π^1 and Π^2 are isomorphic.*

Proof. Let $t^i = v_1^i v_2^i \dots v_{2m}^i$ be the fixed facial walk giving W^i for $i = 1, 2$. We show that the mapping σ given by $\sigma(v_i^1) = v_i^2$ for all i is the desired automorphism. Since $w_i^1 = w_i^2$ for all i, the map σ is a well-defined bijection. Let $\Pi' = \sigma(\Pi)$. We argue that that σ is an automorphism of G as follows: The vertices u and v are adjacent if and only if they are consecutive in t^1. Since the vertex gap vectors W^1 and W^2 are identical, u and v are adjacent if and only if $\sigma(u)$ and $\sigma(v)$ are consecutive in $\sigma(t^1)$, which in turn is if and only if $\sigma(u)$ and $\sigma(v)$ are adjacent in Π'. It is well known that vertices u, v, and w form subwalk uvw of a face F of an embedding Π if and only if w follows u in the local rotation Π_v. Similarly as above, we get that uvw is a subwalk of t^1 if and only if $\sigma(u)\sigma(v)\sigma(w)$ is a subwalk of t^2 and thus w follows u in the local rotation Π_v^1 if and only if $\sigma(w)$ follows $\sigma(u)$ in the local rotation Π', which concludes the proof.

Corollary A.1. *If $W = \mathtt{GAP}(t)$ for the trace corresponding to the unique face of some 1-face embedding Π, then, up to isomorphism, all strong traces t' with $W = \mathtt{GAP}(t')$ correspond to an embedding isomorphic to Π.*

The embedding from Corollary A.1 is called the *associated embedding of W*.

In the following lemma, which is easy to see, by Π^R we denote the embedding obtained by reversing all the local rotations of Π.

Lemma A.2. *Let t be a trace corresponding to the face of a 1-face embedding Π, Then Π^R is a 1-face embedding and t^R is the trace corresponding to the face of Π^R.*

The final ingredients of the proof of Theorem 3.2 is the following fact.

Lemma A.3. *Let $W = \mathtt{GAP}(t)$ for some antiparallel strong trace t corresponding to an embedding Π. Then Π is the associated embedding of all shifts of W.*

Proof. Let W' be the shift of W by one position to the left. Let $t = v_1 \dots v_{2m}$ and let $t' = v_2 \dots v_{2m} v_1$. The definition of the vertex gap representation implies that $W' = \mathtt{GAP}(t')$. Since t and t' are just a shifts of the same unique face of Π, we get that Π is the associated embedding of W' and the result follows by induction.

In the proof of Theorem 3.2 by $t \preceq t'$ we indicate that either $t \prec t'$, or $t = t'$.

Proof. [Proof of Theorem 3.2] First we show that if Π_1 and Π_2 are isomorphic, then $\mathtt{CanonGAP}(t_1) = \mathtt{CanonGAP}(t_2)$. Since we consider Π and Π^R isomorphic and there is not necessarily an automorphism of G taking Π to Π^R, we distinguish two cases: either there is an isomorphism σ taking Π_1 to Π_2, or there is an isomorphism σ taking Π_1 to Π_2^R. In both cases we clearly have $\mathtt{GAP}(t) = \mathtt{GAP}(\sigma(t))$ for any trace t. Let W be a vector obtained by any shift of $\mathtt{GAP}(t_1)$. The proof of Lemma A.3 implies that there is a shift t' of t_1 such that $W = \mathtt{GAP}(t')$. Therefore, for any such W we get that $\sigma(t')$, which is a shift of $\sigma(t_1) = t_2$, also satisfies $\mathtt{GAP}(\sigma(t_1)) = W$. Since an analogous property holds in the case when W is the reversal of a shift of $\mathtt{GAP}(t_1)$, we get that all gaps representations realized by shifts and reversals of t_1 can be realized also by shift and reversals of t_2. Consequently, $\text{lex\_min}(\mathtt{GAP}(t_2)) \preceq \text{lex\_min}(\mathtt{GAP}(t_1))$. Since σ^{-1} is an isomorphism taking Π_2 to Π_1 if and only if σ is an isomorphism taking Π_1 to Π_2, we as well get that $\text{lex\_min}(\mathtt{GAP}(t_1)) \preceq \text{lex\_min}(\mathtt{GAP}(t_2))$, which implies that $\text{lex\_min}(\mathtt{GAP}(t_1)) = \text{lex\_min}(\mathtt{GAP}(t_2))$. Clearly, there is an isomorphism taking Π_1 to Π_2 if and only if there is an isomorphism taking Π_1^R to Π_2^R. Therefore, by the same argument as above we can show that $\text{lex\_min}(\mathtt{GAP}(t_1^R)) = \text{lex\_min}(\mathtt{GAP}(t_2^R))$, yielding $\mathtt{CanonGAP}(t_1) = \mathtt{CanonGAP}(t_2)$, as desired. If there is an isomorphism taking Π_1 to Π_2^R, then analogously as above we can show that $\text{lex\_min}(\mathtt{GAP}(t_1)) = \text{lex\_min}(\mathtt{GAP}(t_2^R))$ and that $\text{lex\_min}(\mathtt{GAP}(t_1^R)) = \text{lex\_min}(\mathtt{GAP}(t_2))$, again yielding $\mathtt{CanonGAP}(t_1) = \mathtt{CanonGAP}(t_2)$.

To prove that if $\mathtt{CanonGAP}(t_1) = \mathtt{CanonGAP}(t_2)$, then Π_1 and Π_2 are isomorphic, consider $\mathtt{CanonGAP}(t_1)$ and $\mathtt{CanonGAP}(t_2)$. Lemma A.3 implies that if both $\mathtt{CanonGAP}(t_1)$ and $\mathtt{CanonGAP}(t_2)$ are shifts of $\mathtt{GAP}(t_1)$, resp. $\mathtt{GAP}(t_2)$, then Π_1 and Π_2 are isomorphic. If both $\mathtt{CanonGAP}(t_1)$ and $\mathtt{CanonGAP}(t_2)$ are reverses of shifts of $\mathtt{GAP}(t_1)$, resp. $\mathtt{GAP}(t_2)$, then Lemma A.3 and Corollary A.1 imply that Π_1^R and Π_2^R are isomorphic, and thus also Π_1 and Π_2 are isomorphic. Assume that exactly one of $\mathtt{CanonGAP}(t_1)$ and $\mathtt{CanonGAP}(t_2)$ is the reverse of a shift of the corresponding gap, say $\mathtt{CanonGAP}(t_1)$. Then Lemma A.3 and Corollary A.1 imply that Π_1^R is isomorphic with Π_2. However, by our definition Π_1 and Π_1^R are isomorphic, which yields the result.

B Additional Figures and Results.

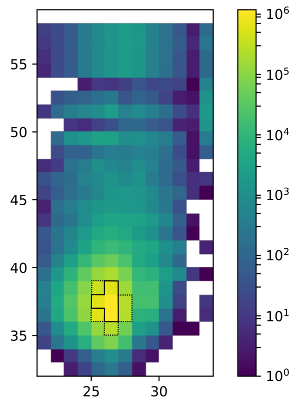
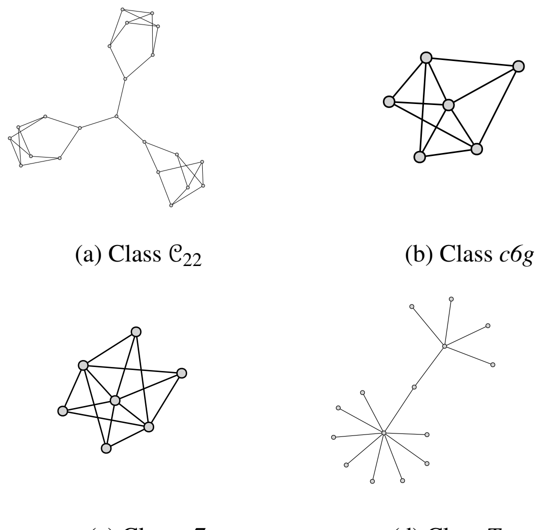


Figure 7: Cubic 22: depicted is a heatplot for the optimal bio-gap (horizontal axis) vs the optimal vertex-gap (vertical axis). The 4 cells surrounded with a bold border cover 50.4% of all graphs, the 9 cells surrounded with a dashed edge cover 74.8% of all graphs



(a) Class $\mathfrak{C}_{22}$ (b) Class *c6g*

(c) Class *c7g* (d) Class T_{15}

Figure 8: a) Of all cubic graphs with 22 nodes the depicted graph has 38 1-face embeddings, no other cubic graph with 22 nodes has fewer; b) depicted is the only graph with 6 nodes (class *c6g*), edge connectivity larger or equal to 3, and a bio gap 6. No other graph with 6 nodes and an edge connectivity larger or equal to 3 has a smaller bio gap; c) depicted is the only graph with 7 nodes (class *c7g*), edge connectivity larger or equal to 3, and a bio gap 6, no other graph with 7 nodes and an edge connectivity larger or equal to 3 has a smaller bio gap; d) depicted is an example graph (a tree with 15 nodes) with $8! \cdot 4! = 967680$ automorphisms, on graph classes with an exponentially growing number of automorphisms algorithm GapEST outperforms AEST

ID																		comment
1	vertex strong trace t	0	2	1	3	0	4	1	2	4	3	1	4	2	0	3	4	
	edge strong trace t_E	02	21	13	30	04	41	12	24	43	31	14	42	20	03	34	40	
	vertex gap repr. $\texttt{GAP}(t)$	4	6	4	6	***9***	3	4	5	3	5	8	4	5	3	5	6	
	$\texttt{GAP}(t^{-1})$	4	5	***9***	5	3	4	6	3	6	4	6	4	5	8	5	3	
	canonical repr. $\texttt{CanonGAP}(t)$	3	4	5	3	5	8	4	5	3	5	6	4	6	4	6	***9***	5-shift of $\texttt{GAP}(t)$
	edge gap repr. $\texttt{EGAP}(t_E)$	***12***	5	7	10	11	5	11	4	6	9	11	***12***	4	6	10	5	
	$\texttt{EGAP}(t_E^{-1})$	6	10	***12***	4	5	7	10	***12***	5	11	5	6	9	11	4	11	
	biological gap repr. $\texttt{BioGAP}(t_E)$	4	5	***7***	6	5	5	5	4	6	***7***	5	4	4	6	6	5	
2	vertex strong trace t	0	2	1	3	0	4	3	1	4	2	0	3	4	1	2	4	
	edge strong trace t_E	02	21	13	30	04	43	31	14	42	20	03	34	41	12	24	40	
	vertex gap repr. $\texttt{GAP}(t)$	4	***8***	5	3	6	3	5	6	4	5	6	***8***	3	5	3	6	
	$\texttt{GAP}(t^{-1})$	3	5	6	4	5	6	***8***	3	5	3	6	4	***8***	5	3	6	
	canonical repr. $\texttt{CanonGAP}(t)$	3	5	3	6	4	***8***	5	3	6	3	5	6	4	5	6	***8***	7-shift of $\texttt{GAP}(t^{-1})$
	edge gap repr. $\texttt{EGAP}(t_E)$	9	***12***	4	7	11	6	***12***	5	6	7	9	10	11	4	10	5	
	$\texttt{EGAP}(t_E^{-1})$	6	***12***	5	6	7	9	10	11	4	10	5	9	***12***	4	7	11	
	biological gap repr. $\texttt{BioGAP}(t_E)$	***7***	4	4	***7***	5	6	4	5	6	***7***	***7***	6	5	4	6	5	
3	vertex strong trace t	0	2	1	3	4	1	2	4	3	0	4	2	0	3	1	4	
	edge strong trace t_E	02	21	13	34	41	12	24	43	30	04	42	20	03	31	14	40	
	vertex gap repr. $\texttt{GAP}(t)$	***9***	5	3	5	3	***9***	5	3	5	3	5	6	4	6	4	5	
	$\texttt{GAP}(t^{-1})$	5	***9***	5	3	5	3	***9***	5	3	5	3	5	6	4	6	4	
	canonical repr. $\texttt{CanonGAP}(t)$	3	5	3	5	6	4	6	4	5	9	5	3	5	3	***9***	5	7-shift of $\texttt{GAP}(t)$
	edge gap repr. $\texttt{EGAP}(t_E)$	11	4	11	4	10	***12***	4	***12***	4	6	***12***	5	***12***	5	6	10	
	$\texttt{EGAP}(t_E^{-1})$	10	11	4	11	4	10	***12***	4	***12***	4	6	***12***	5	***12***	5	6	
	biological gap repr. $\texttt{BioGAP}(t_E)$	5	4	5	4	***6***	4	4	4	4	***6***	4	5	4	5	***6***	***6***	
4	vertex strong trace t	0	2	1	4	2	0	3	4	1	3	0	4	3	1	2	4	
	edge strong trace t_E	02	21	14	42	20	03	34	41	13	30	04	43	31	12	24	40	
	vertex gap repr. $\texttt{GAP}(t)$	5	3	6	4	***10***	5	3	4	5	3	6	4	***10***	5	3	4	
	$\texttt{GAP}(t^{-1})$	4	***10***	5	3	4	5	3	6	4	***10***	5	3	4	5	3	6	
	canonical repr. $\texttt{CanonGAP}(t)$	3	4	5	3	6	4	***10***	5	3	4	5	3	6	4	***10***	5	6-shift of $\texttt{GAP}(t)$
	edge gap repr. $\texttt{EGAP}(t_E)$	4	***12***	5	11	***12***	4	5	11	4	***12***	5	11	***12***	4	5	11	
	$\texttt{EGAP}(t_E^{-1})$	11	***12***	4	5	11	4	***12***	5	11	***12***	4	5	11	4	***12***	5	
	biological gap repr. $\texttt{BioGAP}(t_E)$	4	4	***5***	***5***	4	4	***5***	***5***	4	4	***5***	***5***	4	4	***5***	***5***	

Table 4: Given are the four possible 1-face embeddings of the square pyramid and their corresponding strong traces, the biological-, vertex-, and edge gap representations, and the canonical representations. For edge representations we use the notation ij or ji to indicate the edge $\{i, j\}$. Numbers in bold are the maximum value of the corresponding representation. For the first (resp. second, third and forth) 1-face embedding it holds $\max \texttt{BioGAP}(t) = 7$ (resp. 7,6,5), therefore the fourth 1-face embedding is the only $\texttt{BioGAP}$-optimal. Similarly, the second 1-face embedding is (vertex-)$\texttt{GAP}$-optimal with a gap value of 8, all four 1-face embeddings are $\texttt{EGAP}$-optimal with a gap value of 12.