A Practical Method for the Minimum Genus of a Graph: Models and Experiments

Stephan Beyer[1], Markus Chimani[1⋆], Ivo Hedtke[1], and Michal Kotrbčík[2]

[1] Institute of Computer Science, University of Osnabrück, Germany,
{stephan.beyer,markus.chimani,ivo.hedtke}@uni-osnabrueck.de
[2] Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark, kotrbcik@imada.sdu.dk

Abstract. We consider the problem of the minimum genus of a graph, a fundamental measure of non-planarity. We propose the first formulations of this problem as an integer linear program (ILP) and as a satisfiability problem (SAT). These allow us to develop the first working implementations of general algorithms for the problem, other than exhaustive search. We investigate several different ways to speed-up and strengthen the formulations; our experimental evaluation shows that our approach performs well on small to medium-sized graphs with small genus, and compares favorably to other approaches.

1 Introduction

We are concerned with the minimum genus problem, i.e., finding the smallest g such that a given graph $G = (V, E)$ has an embedding in the orientable surface of genus g. As one of the most important measures of non-planarity, the minimum genus of a graph is of significant interest in computer science and mathematics. However, the problem is notoriously difficult from the theoretical, practical, and also structural perspective. Indeed, its complexity was listed as one of the 12 most important open problems in the first edition of Garey and Johnson's book [22]; Thomassen established its NP-completeness in general [36] and for cubic graphs [37]. While the existence of an $\mathcal{O}(1)$-approximation can currently not be ruled out, there was no general positive result beyond a trivial $\mathcal{O}(|V|/g)$-approximation until a recent breakthrough by Chekuri and Sidiropoulos [9]. For graphs with bounded degree, they provide an algorithm that either correctly decides that the genus of the graph G is greater than g, or embeds G in a surface of genus at most $g^{\mathcal{O}(1)} \cdot (\log |V|)^{\mathcal{O}(1)}$. Very recently, Kawarabayashi and Sidiropoulos [27] showed that the bounded degree assumption can be omitted for the related problem of *Euler genus* by providing a $\mathcal{O}\big(g^{256}(\log |V|)^{189}\big)$-approximation; however, this does not yield an approximation for orientable genus.

Minimum genus is a useful parameter in algorithm design, since, similarly to the planar case, we can take advantage of the topological structure and design faster algorithms for graphs of bounded genus. However, these algorithms typically

⋆ Supported by the German Research Foundation (DFG) project CH 897/2-1

assume that the input graph is actually embedded in some surface, as for instance in [7, 19]. Therefore, without a practical algorithm providing an embedding in a low-genus surface, these algorithms cannot be effectively implemented.

In the mathematical community, the genus of specific graph families is of interest ever since Ringel's celebrated determination of the genus of complete graphs [34]. Such research often combines numerous different approaches, including computer-aided methods, see, e.g., [12, 28]. However, in practice it often turns out that even determining the genus of a single relatively small graph can be rather difficult as in [5, 12, 28, 29, 31]. One of the reasons is the large problem space—an r-regular graph with n vertices can have $[(r-1)!]^n$ embeddings. It is known that complete graphs have exponentially many embeddings of minimum genus; however, the known constructions are nearly symmetric and the problem becomes much more difficult when the minimum genus does not equal the trivial bound from Euler's formula, see, e.g., [28] for more details. While it is conjectured that the genus distribution of a graph—the number of its embeddings into each orientable surface—is unimodal, very little is known about the structure of the problem space both in theory and practice.

From a slightly different perspective, it has been known for a long time that deciding embeddability in a *fixed* surface is polynomial both for the toroidal [20] and the general case [17, 21]. In fact, the minimum genus is fixed-parameter tractable as a result of the Robertson-Seymour theorem, since for every surface there are only finitely many forbidden graph minors, and testing for a fixed minor needs only polynomial time. While there is a direct linear-time algorithm deciding embeddability in a fixed surface [26, 30], taking any of these algorithms to practice is very challenging for several reasons. First, the naïve approach of explicitly testing for each forbidden minor is not viable, since the list of forbidden minors is known only for the plane and the projective plane, and the number of minors grows rapidly: for the torus there are already more than 16 000 forbidden minors [8]. Second, Myrvold and Kocay [33] reviewed existing algorithms to evaluate their suitability for implementation in order to compute the complete list of forbidden toroidal minors. Unfortunately, they report that [20] contains a *"fatal flaw"*, which also appears in the algorithm in [21], and that the algorithm in [17] is also *"incorrect"*. Myrvold and Kocay conclude that *"There appears to be no way to fix these problems without creating algorithms which take exponential time"* [33]. Finally, Mohar's algorithm [30], even in the simpler toroidal case [25], is very difficult to implement correctly (see the discussion in [33]). Consequently, there is currently no correct implementation of any algorithm for the general case of the problem beyond exhaustive search.

It is thus desirable to have an effective and correct implementation of a practical algorithm for the minimum genus. Rather surprisingly, to the best of our knowledge, the approach to obtain practical algorithms via ILP (integer linear program) and SAT (satisfiability) solvers has never been attempted for the minimum genus so far.

Our contribution. We provide the first ILP and SAT formulations for the minimum genus problem, and discuss several different variants both under theoretical and

practical considerations. Based thereon, we develop the first implementations of nontrivial general algorithms for the problem. We evaluate these implementations on benchmark instances widely used in the study of non-planarity measures for real-world graphs. In conjunction with suitable algorithmic support via preprocessing and efficient planarity tests, we are for the first time able to tackle general medium-sized, sparse real-world instances with small genus in practice. We also compare our implementations to existing approaches, namely exhaustive search and a tailored algebraic approach for special cases.

2 Minimum Genus ILP and SAT Formulations

Our terminology is standard and consistent with [32]. We consider finite undirected graphs and assume w.l.o.g. that all graphs are simple, connected, and have minimum degree 3. For each nonnegative integer g, there is, up to homeomorphism, a unique *orientable surface of genus g* and this surface is homeomorphic to a sphere with g added handles. An *embedding* of a graph G in a surface S is a representation of G in S without edge crossings; the minimum genus $\gamma(G)$ of a graph G is the minimum genus of an orientable surface into which G has an embedding. When considering embeddings it is often useful to specify the orientation in which we traverse an edge. Therefore, we may speak of two *arcs* (aka. directed edges, halfedges) that correspond to each edge. For a given graph $G = (V, E)$, let $A = \{uv, vu \mid \{u, v\} \in E\}$ denote the arc set arising from E by replacing each undirected edge by its two possible corresponding directed arcs.

A *rotation* at a vertex v is a cyclic order (counter-clockwise) of the neighbors of v. A *rotation system* of a graph G is a set of rotations, one for each vertex of G. Up to mirror images of the surfaces, there is a 1-to-1 correspondence between rotation systems of G and (cellular) embeddings of G into orientable surfaces (see [23, Thm. 3.2.3] and [18,24]). Given a rotation system of G, the corresponding embedding is obtained by *face tracing*: starting with an unused arc uv, move along it from u to v and continue with the arc vw, where w is the vertex after u at the rotation at v. This process stops by computing a face of the embedding when it re-encounters its initial arc. Repeatedly tracing faces eventually finds all faces of the embedding.

Euler's formula asserts that each (cellular) embedding of G in an orientable surface satisfies $|V| - |E| + f = 2 - 2g$, where f is the number of the faces of the embedding, and g is the genus of the underlying surface. It follows that (i) determining the genus of the underlying surface for a given rotation system is essentially equivalent to calculating the number of faces; and (ii) finding the genus of a graph corresponds to maximizing the number of faces over all rotation systems of the graph. See [32] for more details.

In this section, we describe how to reformulate the minimum genus problem as an integer linear program (ILP) or a related problem of Boolean satisfiability (SAT). Generally, such modeling approaches are known for several planarity concepts and non-planarity measures (e.g., crossing number, graph skewness, upward planarity) and often attain surprisingly strong results. However, for the

minimum genus problem it is at first rather unclear how to capture the topological nature of the question in simple variables. To the best of our knowledge, there are no known formulations for this problem up to now.

We first describe the basic concepts of both formulations, and later consider possible ways to improve them. For convenience, we write $[k] := \mathbb{Z}_k$; addition and subtraction are considered modulo k.

2.1 ILP Formulation

Our formulation is based on finding an embedding with the largest number of faces. Therefore, it statically simulates the face tracing algorithm. Let $\bar{f}$ be an upper bound on the attainable number of faces; see Section 3 on how to obtain a simple linear bound. For each $i \in [\bar{f}]$, we have a binary variable x_i that is 1 iff the i-th face exists and a binary variable c_a^i, for each $a \in A$, that is 1 iff arc a is traversed by the i-th face. For each vertex $v \in V$ and neighbors $u, w \in N(v), u \neq w$, the binary variable $p_{u,w}^v$ is 1 iff w is the successor of u in the rotation at v. The ILP formulation then is:

$$
\begin{aligned}
\max \quad & \textstyle\sum_{i=1}^{\bar{f}} x_i && && \text{(1a)}\\
\text{s.t.} \quad & x_i \leq \tfrac{1}{3} \textstyle\sum_{a \in A} c_a^i && \forall i \in [\bar{f}] && \text{(1b)}\\
& \textstyle\sum_{i=1}^{\bar{f}} c_a^i = 1 && \forall a \in A && \text{(1c)}\\
& \textstyle\sum_{a \in \delta^-(v)} c_a^i = \sum_{a \in \delta^+(v)} c_a^i && \forall i \in [\bar{f}],\ v \in V && \text{(1d)}\\
& c_{vw}^i \geq c_{uv}^i + p_{u,w}^v - 1 && \forall i \in [\bar{f}],\ v \in V,\ u \neq w \in N(v) && \text{(1e)}\\
& c_{uv}^i \geq c_{vw}^i + p_{u,w}^v - 1 && \forall i \in [\bar{f}],\ v \in V,\ u \neq w \in N(v) && \text{(1f)}\\
& \textstyle\sum_{w \in N(v), u \neq w} p_{u,w}^v = 1 && \forall v \in V,\ u \in N(v) && \text{(1g)}\\
& \textstyle\sum_{u \in N(v), w \neq u} p_{u,w}^v = 1 && \forall v \in V,\ w \in N(v) && \text{(1h)}\\
& \textstyle\sum_{u \in U} \sum_{w \in N(v) \setminus U} p_{u,w}^v \geq 1 && \forall v \in V,\ \emptyset \neq U \subsetneq N(v) && \text{(1i)}\\
& x_i \in \{0,1\} && \forall i \in [\bar{f}] && \text{(1j)}\\
& c_a^i \in \{0,1\} && \forall i \in [\bar{f}],\ a \in A && \text{(1k)}\\
& p_{u,w}^v \in \{0,1\} && \forall v \in V,\ u \neq w \in N(v). && \text{(1l)}
\end{aligned}
$$

Constraints (1b) ensure that if a face exists, it traverses *at least* three arcs[3]; inversely, each arc is traversed by exactly one face due to (1c). Equalities (1d) guarantee that at every vertex of a face i, the number of i-traversed incoming and outgoing arcs is identical. Inequalities (1e) and (1f) ensure that arcs uv and vw are both in the same face if w is the successor of u in the rotation at v. Constraints (1g) and (1h) ensure that p^v represents a permutation of the vertices in $N(v)$; (1i) ensures that p^v consists of a single cycle. Observe that maximizing (1a) guarantees that each face index corresponds to at most one facial walk.

[3] For a simple graph, the minimum genus embedding contains no face of length 1 or 2. On the other hand, we cannot be more specific than the lower bound of 3.

2.2 SAT Formulation

To solve the above ILP, we will need to consider its linear relaxation (where the binary variables are replaced by variables in the interval $[0, 1]$). It is easy to see that fractional values for the p^v matrices lead to very weak dual bounds. Therefore, we also consider SAT formulations. While general SAT solvers cannot take advantage of algebraically obtained (lower) bounds, state-of-the-art SAT solvers are highly tuned to quickly search a vast solution space by sophisticated branching, backtracking, and learning strategies. This can give them an upper hand over ILP approaches, in particular when the ILP's relaxation is weak.

In contrast to the ILP, a SAT problem has no objective function and simply asks for *some* satisfying variable assignment. In our case, we construct a SAT instance to answer the question whether the given graph allows an embedding with *at least* f faces. To solve the optimization problem, we iterate the process for increasing values of f until reaching unsatisfiability. We use the same notation as before, and construct the SAT formulation around the very same ideas. Each binary variable is now a Boolean variable instead. While a SAT is typically given in conjunctive normal form (CNF), we present it here as a conjunction of separate Boolean formulae (*rules*) for better readability. Their transformation into equisatisfiable CNFs is trivial. The SAT formulation is:

$$\neg(c_a^i \wedge c_a^j) \qquad \forall a \in A,\, i \neq j \in [f] \tag{2a}$$

$$\bigvee\nolimits_{a \in A}\ c_a^i \qquad \forall i \in [f] \tag{2b}$$

$$p_{u,w}^v \rightarrow (c_{uv}^i \leftrightarrow c_{vw}^i) \qquad \forall v \in V,\, u \neq w \in N(v),\, i \in [f] \tag{2c}$$

$$\bigvee\nolimits_{u \in N(v), u \neq w}\ p_{u,w}^v \qquad \forall v \in V,\, w \in N(v) \tag{2d}$$

$$\neg(p_{u,w}^v \wedge p_{u',w}^v) \qquad \forall v \in V,\, w \in N(v),\, u \neq u' \in N(v) \setminus \{w\} \tag{2e}$$

$$\bigvee\nolimits_{w \in N(v), w \neq u}\ p_{u,w}^v \qquad \forall v \in V,\, u \in N(v) \tag{2f}$$

$$\neg(p_{u,w}^v \wedge p_{u,w'}^v) \qquad \forall v \in V,\, u \in N(v),\, w \neq w' \in N(v) \setminus \{u\} \tag{2g}$$

$$\bigvee\nolimits_{u \in U, w \in N(v) \setminus U}\ p_{u,w}^v \qquad \forall v \in V,\, \emptyset \neq U \subsetneq N(v) \tag{2h}$$

Rules (2a) and (2b) enforce that each arc is traversed by exactly one face, cf. (1c). Rule (2c) ensures that the successor is in the same face, cf. (1e)–(1f). Rules (2d)–(2h) guarantee that p^v variables form rotations at v, cf. (1g)–(1i).

2.3 Improvements

There are several potential opportunities to improve upon the above formulations. In pilot studies we investigated their practical ramifications.

Symmetries (ILP). It seems worthwhile to add symmetry-breaking constraints $x_i \geq x_{i+1}$ or even $\sum_{a \in A} c_a^i \geq \sum_{a \in A} c_a^{i+1}$ for all $i \in [f-1]$ to the ILP. Surprisingly, this does not improve the overall running time (and the latter is even worse by orders of magnitude), and we refrain from using these constraints in the following.

Vertices of degree 3 (ILP&SAT). Let $V_3 := \{v \in V \mid \deg(v) = 3\}$. Consider a degree-3 vertex $v \in V_3$ with neighbors u_0, u_1, u_2. The only two possible rotations at v are $u_0u_1u_2$ and $u_2u_1u_0$. Hence, we can use a single binary/Boolean variable p^v whose assignment represents this choice.

In the ILP, we remove all $p^v_{u,w}$ variables for $v \in V_3$ and replace (1e)–(1i) by

$$c^i_{vu_{k+1}} \geq c^i_{u_kv} + p^v - 1 \quad \forall i \in [f],\, v \in V_3,\, k \in [3] \tag{3a}$$
$$c^i_{u_kv} \geq c^i_{vu_{k+1}} + p^v - 1 \quad \forall i \in [f],\, v \in V_3,\, k \in [3] \tag{3b}$$
$$c^i_{vu_k} \geq c^i_{u_{k+1}v} - p^v \quad \forall i \in [f],\, v \in V_3,\, k \in [3] \tag{3c}$$
$$c^i_{u_{k+1}v} \geq c^i_{vu_k} - p^v \quad \forall i \in [f],\, v \in V_3,\, k \in [3], \tag{3d}$$

where u_0, u_1, u_2 denote the arbitrarily but statically ordered neighbors of $v \in V_3$.

In the SAT formulation, we analogously replace (2c) by

$$p^v \to (c^i_{u_kv} \leftrightarrow c^i_{vu_{k+1}}) \quad \forall v \in V_3,\, k \in [3],\, i \in [f] \tag{4a}$$
$$\neg p^v \to (c^i_{u_{k+1}v} \leftrightarrow c^i_{vu_k}) \quad \forall v \in V_3,\, k \in [3],\, i \in [f]. \tag{4b}$$

As expected, this is faster by orders of magnitude for certain families of graphs, especially for instances with many degree-3 vertices. On the real world *Rome* benchmark set (see Section 4), the performance improves by about 10 % for both the ILP and the SAT formulations, compared to their respective formulations with $p^v_{u,w}$ variables.

This idea can be generalized for vertices v of arbitrary degree $d \geq 4$. There are $\varrho := (d-1)!$ different rotations. Instead of using $\mathcal{O}(d^2)$ many variables $p^v_{u,w}$, we introduce $\lceil \log_2 \varrho \rceil$ binary variables and representing the index of the rotation as a binary number. Since this process is coupled with a substantial trade-off of more complicated and weaker constraints, we refrain from using it for $d \geq 4$.

Binary face representations (SAT). Let $i \in [f]$ be a face index, and $\mathbb{B}(i)$ the vector of its binary representation, i.e., $i = \sum_{j=0}^{\ell} 2^j \cdot \mathbb{B}(i)_j$, where $\ell = \lfloor \log_2 f \rfloor$. We define new Boolean variables b^j_a that are true iff arc a is contained in a face i with $\mathbb{B}(i)_j = 1$. In logic formulae, value $\mathbb{B}(i)_j = 1$ is mapped to *true*, 0 to *false*.

By changing the following clauses of the SAT formulation above, we construct a new formulation that asks for a solution with at least f faces, because we do not forbid the usage of binary representations outside of $[f]$.

$$\bigvee\nolimits_{a \in A} \bigwedge\nolimits_{j \in [\ell]} (b^j_a \leftrightarrow \mathbb{B}(i)_j) \quad \forall i \in [f] \tag{2b′}$$
$$p^v_{u,w} \to (b^j_{uv} \leftrightarrow b^j_{vw}) \quad \forall v \in V \setminus V_3,\, u \neq w \in N(v),\, j \in [\ell] \tag{2c′}$$
$$p^v \to (b^j_{u_kv} \leftrightarrow b^j_{vu_{k+1}}) \quad \forall v \in V_3,\, k \in [3],\, j \in [\ell] \tag{4a′}$$
$$\neg p^v \to (b^j_{u_{k+1}v} \leftrightarrow b^j_{vu_k}) \quad \forall v \in V_3,\, k \in [3],\, j \in [\ell] \tag{4b′}$$

This variant achieves a more than 100-fold speedup.

2.4 Exponential vs. Polynomial Size Formulations

Observe that the number of inequalities (1i), or rules (2h) respectively, is exponential in the degree of each vertex v. Therefore, we investigate ways to obtain a polynomial time solution strategy or a polynomially sized formulation.

Efficient Separation. For the ILP we can *separate* violating constraints (also known as row generation) using a well-known separation oracle based on minimum cuts (see, e.g., [13, Sec. 7.4]). While this guarantees that only a polynomial-sized subset of (1i) is used, it is not worthwhile in practice: the separation process requires a comparably large overhead and state-of-the-art ILP solvers offer a lot of speed-up techniques that need to be deactivated to separate constraints on the fly. Overall, this more than doubles the running times compared to a direct inclusion of all (1i), even if we separate only for vertices with large degrees.

Another option is to use different representations for rotation systems. Here we discuss an *ordering* approach and a *betweenness* approach. Both yield polynomial size formulations.

Ordering Reformulation. For the ordering approach we replace the permutation variables with variables that attach vertices to specific positions in the rotation. This is known to be weaker in the realm of ILPs, and we hence concentrate on the SAT formulation. There, we introduce for any $v \in V, u \in N(v)$ a Boolean variable $q^v_{j,u}$ that is true iff u is the j-th vertex in the rotation at v. We do not use the p variables any longer, replace the old permutation rules (2d)–(2h) with rules to ensure that each q^v is a bijective mapping, and change (2c) to $\bigvee_{j\in[\deg(v)]}\left(q^v_{j,u} \wedge q^v_{j+1,w}\right) \rightarrow (c^i_{uv} \leftrightarrow c^i_{vw})$ for all $v \in V$, $u \neq w \in N(v)$, $i \in [f]$. However, the SAT running times thereby increase 50–100-fold.

Betweenness Reformulation. For the betweenness approach we add the variables $r^v_{x,y,z}$ for each triple $x, y, z \in N(v)$. By $r^v_{x,y,z} = 1$ (true, respectively) we denote that y is (somewhere) between x and z in the rotation at v. Here we only describe the usage of the r variables in the SAT formulation. The usage in the ILP is analogous. First of all, the cyclicity of a rotation implies the symmetries $r^v_{x,y,z} \equiv r^v_{y,z,x} \equiv r^v_{z,x,y} \equiv \neg r^v_{x,z,y} \equiv \neg r^v_{z,y,x} \equiv \neg r^v_{y,x,z}$ for all $\{x, y, z\} \subseteq N(v)$. Instead of ensuring that each p^v represents a permutation, we connect the p variables to the new r variables via $p^v_{u,w} \leftrightarrow \bigwedge_{y\in N(v)\setminus\{u,w\}} r^v_{u,w,y}$. The rules to model the betweenness conditions for the neighborhood of a given vertex v are simply $r^v_{u,w,x} \wedge r^v_{u,x,y} \rightarrow r^v_{u,w,y} \wedge r^v_{w,x,y}$ for all $\{u, w, x, y\} \subseteq N(v)$. However, the SAT running times thereby increase 20–50-fold.

Overall, we conclude that the exponential dependencies of the original formulations are not so much of an issue in practice after all, and the overhead and weaknesses of polynomial strategies typically seem not worthwhile. However, if one considers problems with many very high degree vertices where the exponential dependency becomes an issue, the above approaches can be merged very naturally, leading to an overall polynomial model: Let τ be some fixed constant

threshold value (to be decided upon experimentally). For vertices v of degree at most τ, we use the original formulation requiring an exponential (in constant τ) number of constraints over p^v. Vertices of degree above τ are handled via the betweenness reformulation.

3 A Minimum Genus Computation Framework

Before deploying any of our approaches on a given graph, we consider several preprocessing steps. Since the genus is additive over biconnected components [1,2], we decompose the input graph G accordingly. We can test $\gamma = 0$ by simply running a linear time planarity test, in our case [4]. Next, we observe that the genus problem is susceptible to *non-planar core reduction* [10]: A *maximal planar 2-component* is defined as a maximal subgraph $S \subset G$ that (i) has only two vertices x, y in common with the rest of the graph, and (ii) $S + (x, y)$ is planar. The (in our case unweighted) non-planar core (NPC) of G is obtained (in linear time) by replacing each such maximal planar 2-components by an edge.[4] After these steps we are in general left with a set of simple biconnected (preprocessed) graphs with minimum degree at least 3, for each of which we want to compute the genus.

By Euler's formula, we only have to calculate SAT instances with $f \equiv |E|-|V| \mod 2$. For increasing number of faces we compute the satisfiability until we get the first unsatisfiable instance. Such an iteration is clearly not necessary in the ILP approach, where our objective function explicitly maximizes f and we only require an upper bound of $\bar{f} = \min\{\lfloor 2|E|/3 \rfloor, |E| - |V|\}$,[5] adjusted for parity.

4 Experimental Evaluation

Our C++ code is compiled with GCC 4.9.2, and runs on a single core of an AMD Opteron 6386 SE with DDR3 Memory @ 1600 MHz under Debian 8.0. We use the ILP solver CPLEX 12.6.1, the SAT solver lingeling (improved version for SMT Competition 2015 by Armin Biere)[6], and the Open Graph Drawing Framework (www.ogdf.net, GPL), and apply a 72 GB memory limit.

[4] In [10], the validity of such a preprocessing is shown for several non-planarity measures, namely crossing number, skewness, coarseness, and thickness. Let H be the NPC of G. We can trivially observe that (A) $\gamma(G) \leq \gamma(H)$, and (B) $\gamma(G) \geq \gamma(H)$. A: Given an optimal solution for H, we can embed each S onto the surface in place of its replacement edge, without any crossings. B: Each replaced component S contains a path connecting its poles that is drawn crossing-free in the optimal embedding of G; we can planarly draw all of S along this path, and then simplify the embedding by replacing this locally drawn S by its replacement edge; this gives a solution for H on the same surface.

[5] First term: each edge lies on at most two faces, each face has size at least 3; second term: Euler's formula with genus at least 1.

[6] The previous version was the winner of the *Sequential Appl. SAT+UNSAT Track* of the SAT competition 2014 [3]. This improved version is even faster.

Table 1. Characteristics of instances and resulting formulations. The graphs from the *Rome* (left table) and *North* (right table) benchmark sets are grouped by their number of vertices in the given ranges. For each group, we give the averages for the following values: number of vertices and percentage of degree-3 vertices in the NPC, upper bound $\bar{f}$ on the number of faces, number of variables and constraints in the ILP formulation.

range	avg. for computation on NPC				
$\|V\|$	$\|V\|$	$\%\|V_3\|$	$\bar{f}$	#vars	#cons
10–40	12.8	64.2	10.0	616.1	3399.5
41–60	18.5	60.3	15.3	1310.7	7639.9
61–80	26.8	59.4	22.5	2624.4	15735.1
81–100	36.4	58.5	30.9	4718.4	28778.3

range	avg. for computation on NPC				
$\|V\|$	$\|V\|$	$\%\|V_3\|$	$\bar{f}$	#vars	#cons
10–40	12.6	38.3	17.4	2200.0	102295.9
41–60	24.6	40.3	29.9	4916.7	197577.3
61–80	32.1	43.5	35.5	7741.7	249864.6
81–100	24.3	40.6	34.7	7146.7	632634.6

Real world graphs. We consider the established *Rome* [16] and *North* [15] benchmark sets of graphs collected from real-world applications. They are commonly used in the evaluation of algorithms in graph drawing and non-planarity measures. We use the ILP and SAT approaches to compute the genera of all 8249 (423) non-planar Rome (North) graphs. Each approach is run with a 30 minute time limit for each graph to compute its genus; we omit 10 (*North*) instances that failed due to the memory limitation. Characteristics about the data sets and the resulting formulations can be found in Table 1.

Fig. 1(a) shows the success rate (computations finished within the time limit) for the Rome graphs, depending on the number of vertices of the input graph. Both the SAT and ILP approach exhibit comparable numbers, but nearly always, the success rate of the SAT approach is as good or better than the ILP's. However, the differences are statistically not significant. Instances with up to 40 vertices can be solved with a high success rate; our approach degrades heavily for graphs with more than 60–70 vertices. However, it is worth noting that even if the genus is not calculated to provable optimality, we obtain highly nontrivial bounds on the genus of the graphs in question.

In Fig. 1(b) we see that, given *any* fixed time limit below 30 minutes, the SAT approach solves clearly more instances than the ILP approach. Note that the curve that corresponds to the solved SAT instances flattens out very quickly.

When we compare the success rates to the density of the NPC (see Fig. 1(c)), we see the same characteristics as in Fig. 1(a). Both approaches are able to solve instances with density (i.e., $|E|/|V|$) up to 1.6 with a high success rate but are typically not able to obtain provably optimal values for densities above 1.9.

Finally, we compare the average running time of the instances that are solved by both approaches. Out of the 8249 non-planar Rome graphs we are able to solve 2571 with SAT *and* ILP, and additionally 96 (24) more with the SAT (ILP, respectively). Except for very small graphs, the average running time of the SAT approach is always at least one or two orders of magnitude lower than the average running time of the ILP approach, see Fig. 1(d).

Considering the non-planar North graphs, Fig. 1(e) shows that the success rates of both approaches are again comparable. Again, the differences are sta-

tistically not significant. However, ten instances could not be solved due to the high memory consumption caused by the exponential number of constraints (1i) and rules (2h). Since the results for the North graphs are analogous to those for the Rome graphs, we omit discussing them in detail.

Generally, we observe that the SAT approach is particularly fast to show the existence of an embedding, but is relatively slow to prove that there is no embedding with a given number of faces. This is of particular interest for non-planar graphs that allow a genus-1 embedding, since there the SAT is quick to find such a solution and need not prove that a lower surface is infeasible. The SAT's behavior in fact suggests an easy heuristical approach: if solving the SAT instance for f faces needs a disproportionally long running time (compared to the previous iterations for lower face numbers), this typically indicates that it is an unsatisfiable instance and $f - 2$ faces is the optimal value.

Comparison to existing genus computations. An evaluation of exhaustive search algorithms for determining the genus of complete graphs was performed in [35]. Fixing the rotation of the first vertex, it is possible to compute the genus of the complete graph K_7 within 896 hours of computation (112 hours on 8 parallel threads). While *both* our approaches perform significantly better, there is a notable (and w.r.t. to the above evaluations particularly surprising) difference in their performance: the SAT approach needs 1 hour to find and prove the optimal genus; solving the ILP takes only 30 seconds.

A circulant $C_n(S)$ is the Cayley graph of $\mathbb{Z}_n$ with generating set S. Conder and Grande [12] recently characterized all circulants with genus 1 and 2. A crucial part of the characterization is the determination of the genus of several sporadic cases where the lower bounds are more problematic. At the same time, these sporadic cases constitute the main obstacle in both obtaining a simpler proof, as well as extending the results to higher genera. By far the most difficult case is proving that the genus of $C_{11}(1, 2, 4)$ is at least 3. The proof takes three pages of theoretical analysis and eventually resorts to a computational verification of three subcases, taking altogether around 85 hours using the MAGMA computational algebra system in a nontrivial problem-specific setting. The ILP solver needs 180 hours to determine the genus without using any theoretical results or problem-specific information.

5 Conclusion

The minimum genus problem is very difficult from the mathematical, algorithmic, and practical perspective—the problem space is large and seems not to be well-structured, the existing algorithms are error-prone and/or very difficult to implement, and only little progress was made on the (practice-oriented) algorithmic side. In this paper we have presented the first ILP and SAT formulations, together with several variants and alternative reformulations, for the problem, and investigated them in an experimental study. Our approach leads to the first (even easily!) implementable general-purpose minimum genus algorithms.

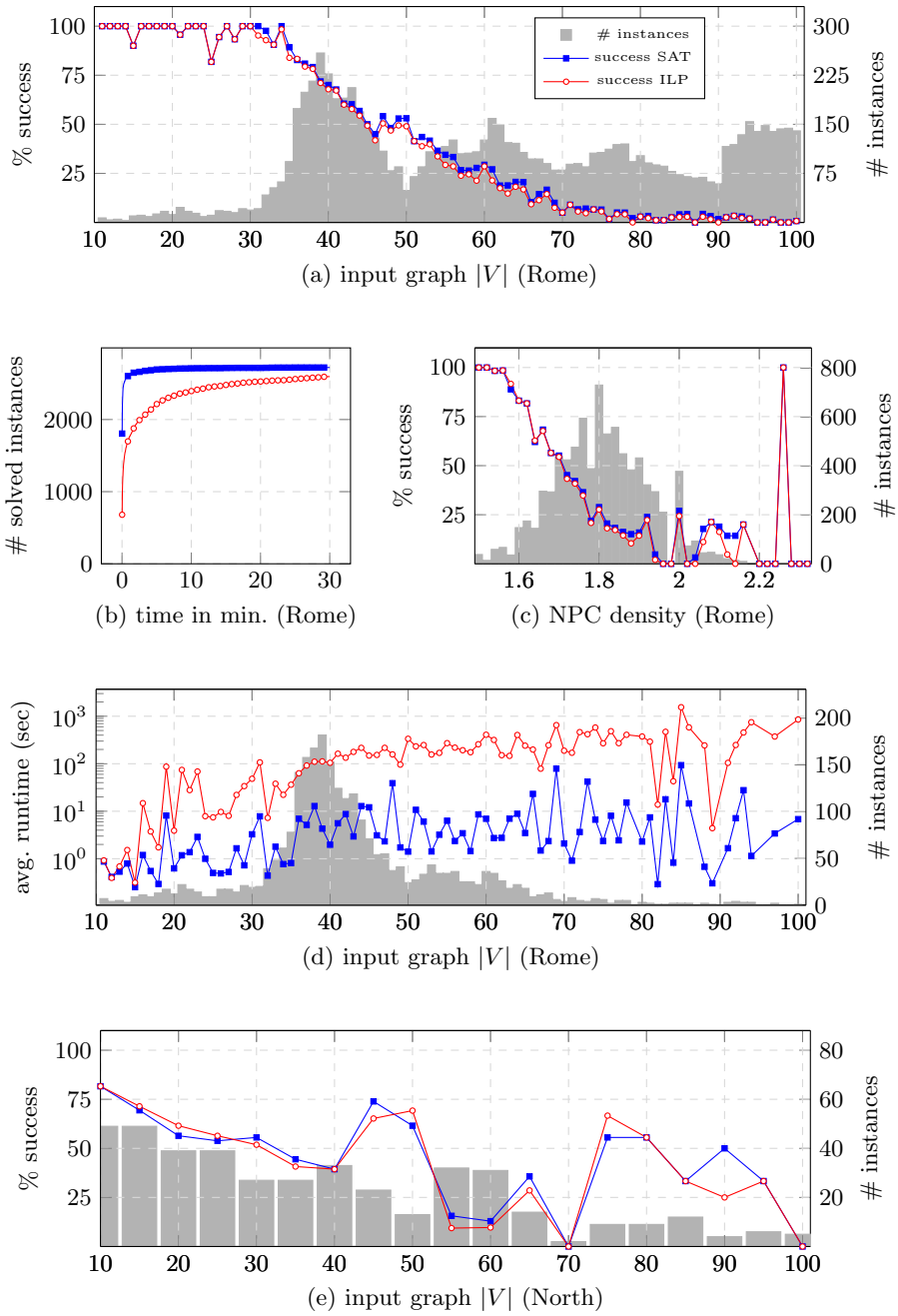

Fig. 1. Rome Graphs: (a) success rate per $|V|$, (b) solved instances per given time, (c) success rate per non-planar core density $|E|/|V|$, (d) average running time per $|V|$ where both approaches were successful. North graphs: (e) success rate per $|V|$.

Besides yielding practical algorithms for small to medium-sized graphs and small genus, one of the further advantages of our approach is that the formulations are adaptable and can be modified to tackle other related problems of interest. For example, the existence of polyhedral embeddings [32], or embeddings with given face lengths, say 5 and 6 as in the case of fullerenes (graph-theoretic models of carbon molecules), see [14].

On the negative side, our implementations cannot deal with too large graphs without resorting to extensive computational resources. However, this is not very surprising considering the difficulty of the problem—a fast exact algorithm could be used to solve several long-standing open problems, such as completing the list of forbidden toroidal minors. We also see—and hope for—certain similarities to the progress on exact algorithms for the well-known crossing number problem: while the first published report [6] was only capable of solving Rome graphs with 30–40 vertices, it led to a series of improvements that culminated in the currently strongest variant [11] which is capable to tackle even the largest Rome graphs.

Acknowledgements. We thank Armin Biere for providing the most recent version (as of 2015-06-05) of the lingeling SAT solver.

References

1. Archdeacon, D.: The orientable genus is nonadditive. J. Graph Theory 10(3), 385–401 (1986)
2. Battle, J., Harary, F., Kodama, Y., Youngs, J.W.T.: Additivity of the genus of a graph. Bull. Amer. Math. Soc. 68, 565–568 (1962)
3. Belov, A., Diepold, D., Heule, M.J., Järvisalo, M. (eds.): Proceedings of SAT Competition 2014: Solver and Benchmark Descriptions. No. B-2014-2 in Series of Publications B, University of Helsinki, Dept. Comp. Sc. (2014)
4. Boyer, J.M., Myrvold, W.J.: On the cutting edge: Simplified $O(n)$ planarity by edge addition. J. Graph Algorithms Appl. 8(2), 241–273 (2004)
5. Brin, M.G., Squier, C.C.: On the genus of $Z_3 \times Z_3 \times Z_3$. European J. Combin. 9(5), 431–443 (1988)
6. Buchheim, C., Ebner, D., Jünger, M., Klau, G.W., Mutzel, P., Weiskircher, R.: Exact crossing minimization. In: Proc. GD '05. pp. 37–48. LNCS 3843 (2005)
7. Cabello, S., Chambers, E.W., Erickson, J.: Multiple-source shortest paths in embedded graphs. SIAM J. Comput. 42(4), 1542–1571 (2013)
8. Chambers, J.: Hunting for torus obstructions. M.Sc. thesis, Uni. of Victoria (2002)
9. Chekuri, C., Sidiropoulos, A.: Approximation algorithms for euler genus and related problems. In: Proc. FOCS '13. pp. 167–176 (2013)
10. Chimani, M., Gutwenger, C.: Non-planar core reduction of graphs. Disc. Math. 309(7), 1838–1855 (2009)
11. Chimani, M., Mutzel, P., Bomze, I.: A new approach to exact crossing minimization. In: Proc. ESA '08. pp. 284–296. LNCS 5193 (2008)
12. Conder, M., Grande, R.: On embeddings of circulant graphs. Electronic J. Combin. 22(2), P2.28 (2015)
13. Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A.: Combinatorial Optimization. Wiley-Interscience Ser. in Discr. Mathematics and Optimization (1998)

14. Deza, M., Fowler, P.W., Rassat, A., Rogers, K.M.: Fullerenes as tilings of surfaces. J. Chemical Information and Computer Sciences 40(3), 550–558 (2000)
15. Di Battista, G., Garg, A., Liotta, G., Parise, A., Tamassia, R., Tassinari, E., Vargiu, F., Vismara, L.: Drawing directed acyclic graphs: An experimental study. Int. J. Comput. Geometry Appl. 10(6), 623–648 (2000)
16. Di Battista, G., Garg, A., Liotta, G., Tamassia, R., Tassinari, E., Vargiu, F.: An experimental comparison of four graph drawing algorithms. Computational Geometry 7(5–6), 303–325 (1997)
17. Djidjev, H., Reif, J.: An efficient algorithm for the genus problem with explicit construction of forbidden subgraphs. In: Proc. STOC '91. pp. 337–347. ACM (1991)
18. Edmonds, J.: A combinatorial representation for polyhedral surfaces. Notices Amer. Math. Soc. 7, 646 (1960)
19. Erickson, J., Fox, K., Nayyeri, A.: Global minimum cuts in surface embedded graphs. In: Proc. SODA '12. pp. 1309–1318. SIAM (2012)
20. Filotti, I.S.: An efficient algorithm for determining whether a cubic graph is toroidal. In: Proc. STOC '78. pp. 133–142. ACM (1978)
21. Filotti, I.S., Miller, G.L., Reif, J.: On determining the genus of a graph in $O(V^{O(G)})$ steps. In: Proc. STOC '79. pp. 27–37. ACM (1979)
22. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the theory of NP-completeness. Bell Telephone Laboratories (1979)
23. Gross, J.L., Tucker, T.W.: Topological graph theory. Wiley-Interscience Series in Discrete Mathematics and Optimization (1987)
24. Heffter, L.: Ueber das Problem der Nachbargebiete. Math. Ann. 38, 477–508 (1891)
25. Juvan, M., Marinček, J., Mohar, B.: Embedding graphs in the torus in linear time. In: Proc. IPCO '95. pp. 360–363. LNCS 920 (1995)
26. Kawarabayashi, K., Mohar, B., Reed, B.: A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In: Proc. FOCS '08. pp. 771–780 (2008)
27. Kawarabayashi, K.i., Sidiropoulos, A.: Beyond the euler characteristic: Approximating the genus of general graphs. In: Proc. STOC '15. ACM (2015)
28. Kotrbčík, M., Pisanski, T.: Genus of cartesian product of triangles. Electronic J. Combin. 22(4), P4.2 (2015)
29. Marušič, D., Pisanski, T., Wilson, S.: The genus of the GRAY graph is 7. European J. Combin. 26(3–4), 377–385 (2005)
30. Mohar, B.: Embedding graphs in an arbitrary surface in linear time. In: Proc. STOC '96. pp. 392–397. ACM (1996)
31. Mohar, B., Pisanski, T., Škoviera, M., White, A.: The cartesian product of 3 triangles can be embedded into a surface of genus 7. Disc. Math. 56(1), 87–89 (1985)
32. Mohar, B., Thomassen, C.: Graphs on Surfaces. Johns Hopkins Studies in the Mathematical Sciences (2001)
33. Myrvold, W., Kocay, W.: Errors in graph embedding algorithms. Journal of Computer and System Sciences 77(2), 430–438 (2011)
34. Ringel, G.: Map Color Theorem. Springer-Verlag (1974)
35. Schmidt, P.: Algoritmické vlastnosti vnorení grafov do plôch. B.Sc. thesis, Comenius University (2012), in Slovak
36. Thomassen, C.: The graph genus problem is NP-complete. Journal of Algorithms 10, 568–576 (1989)
37. Thomassen, C.: The graph genus problem is NP-complete for cubic graphs. J. Combinatorial Theory ser. B 69, 52–58 (1997)