Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 346 (2019) 185–197

www.elsevier.com/locate/entcs

On Computing the Path Number of a Graph

F. Botler[2]

Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil

R. Cano[3]

Instituto de Computação
Universidade Estadual de Campinas
Campinas, Brazil

M. Sambinelli[4]

Departamento de Ciência da Computação
Universidade de São Paulo
São Paulo, Brazil

Abstract

Gallai (1966) conjectured that the edge set of every graph G on n vertices can be covered by at most $\lceil n/2 \rceil$ edge-disjoint paths. Such a covering by edge-disjoint paths is called a *path decomposition*, and the size of a path decomposition with a minimum number of elements is called the *path number* of G. Peroche (1984) proved that the problem of computing the path number is NP-Complete; and Constantinou and Ellinas (2018) proved that it is polynomial for a family of complete bipartite graphs. In this paper we present an Integer Linear Programming model for computing the path number of a graph. This allowed us to verify Gallai's Conjecture for a large collection of graphs. As a result, following a work of Heinrich, Natale and Streicher on cycle decompositions (2017), we verify Gallai's Conjecture for graphs with at most 11 vertices; for bipartite graphs with at most 16 vertices; and for regular graphs with at most 14 vertices.

Keywords: path decomposition, integer linear programming, small graphs

[1] This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. M. Sambinelli was partially supported by National Counsel of Technological and Scientific Development of Brazil (CNPq), grant 141216/2016-6, and São Paulo Research Foundation (FAPESP), grant 17/23623-4.

[2] Email: fbotler@cos.ufrj.br

[3] Email: rgcano@ic.unicamp.br

[4] Email: sambinelli@ime.usp.br

https://doi.org/10.1016/j.entcs.2019.08.017

1 Introduction

All graphs considered here are finite and simple, i.e., contain a finite number of vertices and edges and have neither loops nor multiple edges. The terminology and notation used here are standard and we refer the reader to Bondy and Murty's book for missing definitions [3]. A *decomposition* $\mathcal{D}$ of a graph G is a collection of edge-disjoint subgraphs of G that covers all the edges of G. A decomposition $\mathcal{D}$ is a *path decomposition* if every element in $\mathcal{D}$ is a path. A path decomposition $\mathcal{D}$ of a graph G is *minimum* if for every path decomposition $\mathcal{D}'$ of G we have $|\mathcal{D}| \leq |\mathcal{D}'|$. Erdős (see [2,21]) suggested the investigation of the cardinality of such a minimum path decomposition, which we call the *path number* of G, and denote by $\mathrm{pn}(G)$. To answer Erdős' question, Gallai posed the following conjecture.

Conjecture 1.1 (Gallai, 1968) *If G is a connected graph on n vertices, then* $\mathrm{pn}(G) \leq \lceil \frac{n}{2} \rceil$.

In the past 50 years, Conjecture 1.1 has been the object of several studies [1,10,11,12,13,20,21,24]. Recently, Conjecture 1.1 was listed by Adrian Bondy [2] as one of the most beautiful open conjectures in Graph Theory. Although Conjecture 1.1 has been explored in many directions and has been verified for several particular cases, it remains open in the general case. Lovász [21] verified Conjecture 1.1 for graphs with at most one vertex with even degree. Pyber [24] extended Lovász's result by proving that Conjecture 1.1 holds for graphs whose cycles have vertices with odd degree. Another step of generalization was given by Fan [10], who verified Gallai's Conjecture in the case G is a graph such that each block of G_E has no triangles and has maximum degree 3, where G_E is the graph obtained from G by removing its vertices with odd degree.

In another direction, Favaron and Kouider [11] verified Conjecture 1.1 for Eulerian graphs with maximum degree at most 4, Jiménez and Wakabayashi [20] verified it for a family of triangle-free graphs, and Botler and Jiménez [4] verified it for a family of even regular graphs with high girth condition. Also, Geng, Fang, and Li [12] verified Conjecture 1.1 for maximal outerplanar graphs and 2-connected outerplanar graphs, and Bonamy and Perrett [1] verified it for graphs with maximum degree at most 5. More recently, Botler, Sambinelli, Coelho, and Lee [6] verified Conjecture 1.1 for graphs with treewidth at most 3 by proving that a partial 3-tree with n vertices is either a *Gallai graph*, i.e., a graph with path number at most $\lfloor n/2 \rfloor$, or is one of the two exceptions (K_3 and $K_5 - e$). They also proved that graphs with maximum degree at most 4 are either Gallai graphs or one of three exceptions (K_3, $K_5 - e$, and K_5). In another work, Botler, Jiménez, and Sambinelli [5] proved that every triangle-free planar graph is a Gallai graph. We note that, although not presented explicitly, the proofs in all these papers are algorithmic, and hence one can obtain, for each case, a polynomial time algorithm that finds a path decomposition of size at most $\lceil n/2 \rceil$. For some other results concerning Conjecture 1.1 we refer the reader to [8,9,13,18].

Frequently, when working on such problems, it is useful to be able to test a hypothesis in a large collection of graphs. Unfortunately, from a purely algorithmic

perspective of computing the path number, not many results are known. Peroche [23] proved that computing the path number of a connected graph is NP-hard even for graphs with maximum degree at most 4; and Constantinou and Ellinas [7] presented a polynomial algorithm to compute the path number of some complete bipartite graphs. The motivation for this work arose from the need of the authors to test such hypothesis for Conjecture 1.1, and this work is the result of such learnings. Our main contribution is an Integer Linear Programming (ILP) formulation to compute the path number of a graph (see Section 2). Moreover, in Section 3, following a work of Heinrich, Natale and Streicher on cycle-decompositions [19], we discuss how we used such ILP together with heuristics and known theoretical results to verify Conjecture 1.1 for 15.871.356.558 graphs, resulting in Theorem 1.2. In Section 4, we present some concluding remarks and discuss future works.

Theorem 1.2 *Let G be a connected graph on n vertices. Then* $\mathrm{pn}(G) \leq \lceil n/2 \rceil$ *if at least one of the following holds. (i) $n \leq 11$; (ii) $n \leq 16$ and G is bipartite; or (iii) $n \leq 14$ and G is regular.*

2 Integer Linear Programming formulation

In this section we present an ILP formulation to compute the path number of a graph. Given a vertex v, we denote by $\delta(v)$ the set of edges of G incident to v. An *angle* in a graph G is a pair $\{e, f\} \subseteq \delta(v)$ of edges incident to a common vertex $v \in V(G)$; in this case, we say that v *supports* the angle $\{e, f\}$. We denote the set of all angles of G by $\mathcal{A}(G)$. Alternatively, $\mathcal{A}(G)$ corresponds to the set of edges of the line graph $L(G)$ of G. We denote by $\mathcal{C}_G$ the set of all simple cycles of G. Note that, if C is a simple cycle, then $\mathcal{A}(C)$ is the set of pairs of consecutive edges of C. Our formulation is presented in Model 1.

For each angle $\{e, f\} \in \mathcal{A}(G)$, we define a binary variable x_{ef} that is set to 1 only if e and f are (consecutive) edges of the same element of a path decomposition. The objective function maximizes the number of angles in the solution. We show this to be equivalent to minimizing the cardinality of a path decomposition. Inequalities (1b) state that in a solution no two angles supported by a same vertex have a common edge (omitting it allows the elements of its corresponding decomposition to be walks that are not trails). Inequalities (1c), called the *cycle inequalities*, guarantee that no element of the induced decomposition contains cycles. We can show that the polytope associated with Model 1 is full-dimensional. Moreover, Inequalities (1b) are facet-defining when applied to vertices of degree at least 4, and Inequalities (1c) are always facet-defining. Due to space limitations we omit some proofs.

Now, given a solution x^* to Model 1, we construct a path decomposition $\mathcal{D}^* = \mathcal{D}(x^*)$ of G. Let $\mathcal{D}_0 = \big\{G[e] \colon e \in E(G)\big\}$, i.e., $\mathcal{D}_0$ is the path decomposition of G where each path contains a single edge, and let $\mathcal{A}_1 = \big\{\{e_i, f_i\} \in \mathcal{A}(G) \colon x^*_{e_i f_i} = 1\big\} = \big\{\{e_1, f_1\}, \{e_2, f_2\}, \ldots, \{e_k, f_k\}\big\}$. Given an edge $e \in E(G)$ and a decomposition $\mathcal{B}$ of G, we denote by $\mathcal{B}^e$ the element in $\mathcal{B}$ containing the edge e. The decomposition $\mathcal{D}^*$ is then obtained from D_0 by successively joining elements as follows. For $i = 1, \ldots, k$,

$$\max \quad \mu(x) = \sum_{\{e,f\}\in\mathcal{A}(G)} x_{ef} \tag{1a}$$

$$\text{s.t.} \quad \sum_{f\in\delta(v)\setminus\{e\}} x_{ef} \leq 1 \qquad \forall v \in V(G), \forall e \in \delta(v) \tag{1b}$$

$$\sum_{\{e,f\}\in\mathcal{A}(C)} x_{ef} \leq |E(C)| - 2 \qquad \forall C \in \mathcal{C}_G \tag{1c}$$

$$x_{ef} \in \{0,1\} \qquad \forall \{e,f\} \in \mathcal{A}(G) \tag{1d}$$

Model 1: An ILP formulation for computing the path number of a graph.

let $\mathcal{D}_i = \left(\mathcal{D}_{i-1} \setminus \left\{\mathcal{D}_{i-1}^{e_i}, \mathcal{D}_{i-1}^{f_i}\right\}\right) \cup \left\{\mathcal{D}_{i-1}^{e_i} \cup \mathcal{D}_{i-1}^{f_i}\right\}$, and put $\mathcal{D}^* = \mathcal{D}_k$. It is not hard to check that Inequality (1b) implies that every element in $\mathcal{D}^*$ is a trail, i.e., given an edge $e \in E(G)$, Inequality (1b) defines the edges of G that are neighbors of e in $\mathcal{D}^*$. Moreover, for every element $T \in \mathcal{D}^*$ there is an ordering $S_T = v_0 v_1 \cdots v_k$ of its vertices such that for every pair $\{uv, vw\} \in \mathcal{A}_1$ with $uv, vw \in E(T)$, either uvw or wvu is a subsequence of S_T. Finally, let $T \in \mathcal{D}^*$ and S_T be the ordering above. If T is not a path, then there is a closed subsequence $S' = v_i v_{i+1} \cdots v_j$ of S_T in which the only vertex that appears twice is $v_i = v_j$. Then S' induces a cycle of length $j-i$ in G. Moreover, we have $\{v_l v_{l+1}, v_{l+1} v_{l+2}\} \in \mathcal{A}_1$ for every $l \in \{i, \ldots, j-2\}$. But this implies $\sum_{\{e,f\}\in\mathcal{A}(S')} x^*_{ef} \geq j-i-1 > |E(S')|-2$, a contradiction to Inequality (1c). Finally, since the join operation corresponding to each angle in $\mathcal{A}_1$ decreases the size of the decomposition by 1, we have

$$|\mathcal{D}(x^*)| = |\mathcal{D}^*| = |E(G)| - k = |E(G)| - \sum_{\{e,f\}\in\mathcal{A}(G)} x^*_{ef} = |E(G)| - \mu(x^*). \tag{2}$$

Theorem 2.1 *Let x^* be a feasible solution of Model 1 for a graph G. Then x^* is an optimal solution if and only if $\mathcal{D}(x^*)$ is a minimal path decomposition of G.*

Proof Let $\mathcal{D}$ be a minimum path decomposition of G. For each $\{e,f\} \in \mathcal{A}(G)$, let $\widehat{x}_{ef} = 1$ if e and f are consecutive edges in some path of $\mathcal{D}$, and let $\widehat{x}_{ef} = 0$, otherwise. It is not hard to check that $\widehat{x}$ satisfies Inequalities (1b), (1c), and (1d) and that $\mu(\widehat{x}) = |E(G)| - |\mathcal{D}|$. Therefore, it follows from (2) that

$$\mu(x^*) = |E(G)| - |\mathcal{D}(x^*)| \leq |E(G)| - |\mathcal{D}| = \mu(\widehat{x}) \leq \mu(x^*).$$

□

2.1 Additional inequalities

In this section, we present additional valid inequalities to strengthen our formulation. Let G be a graph on n vertices. We denote by n_{odd}, the number of vertices with odd degree in G, and by $\Delta(G)$, the maximum degree of G. Given a path decomposition $\mathcal{D}$ of G and a vertex $v \in V(G)$, we denote by $\mathcal{D}(v)$ the number of paths of $\mathcal{D}$ that have v as an end vertex. It is not hard to check that we have $\mathcal{D}(v) \equiv d(v) \pmod 2$ for every vertex of G. Therefore, $\mathcal{D}(v) \geq 1$ for every vertex of odd degree. This implies that $|\mathcal{D}| \geq n_{\text{odd}}/2$. Also, note that, since a path contains at most two edges incident to any vertex, there must be at least $\lceil d(v)/2 \rceil$ paths containing vertex v, and hence, $\mathcal{D}$ has at least $\lceil \Delta(G)/2 \rceil$ elements. Moreover, since any path in G contains at most $n-1$ edges, $\mathcal{D}$ has at least $\lceil |E(G)|/(n-1) \rceil$ elements. These observations can be summarized in the following inequality.

$$|\mathcal{D}| = |E(G)| - \sum_{\{e,f\} \in \mathcal{A}(G)} x_{ef} \geq \left\lceil \max \left\{ \frac{n_{\text{odd}}}{2}, \frac{\Delta(G)}{2}, \frac{|E(G)|}{|V(G)|-1} \right\} \right\rceil. \tag{3}$$

Let S_n be a star with n edges, i.e., S_n is the complete bipartite graph $K_{1,n}$, and let v be its center vertex. Since each path of a decomposition $\mathcal{D}$ contains at most two edges incident to v, each path may set at most one angle supported by v. Also, since $\mathcal{D}$ is a decomposition, each edge incident to v is in exactly one path of $\mathcal{D}$, and hence it is in at most one angle of v. Therefore, we have $\sum_{\{e,f\} \in \mathcal{A}(S_n)} x_{ef} \leq \lfloor n/2 \rfloor$. It can be shown that, when n is even, this inequality is implied by Inequality (1b). However, when n is odd, we obtain a stronger inequality, which we refer to as a *star inequality*.

$$\sum_{\{e,f\} \in \mathcal{A}(S_{2k+1})} x_{ef} \leq k \qquad \forall S_{2k+1} \subseteq G. \tag{4}$$

It can be shown that Inequalities (4) are always facet-defining for any subgraph of G consisting of a star with an odd number of edges. In particular, for vertices with degree 3, it dominates Inequality (1b). As for Inequality (3), we currently do not know its dimension, although it seems very effective in practice.

2.2 Branching strategy

When experimenting with Model 1 using CPLEX, we observed that the solver's default branching choices were not effective and led to the exploration of a large number of nodes. Therefore, we propose a different branching strategy. As we observe in Section 3, when dealing with dense graphs, it is often useful to select and remove one or more paths and continue solving the problem with a sparser graph. Naturally, this idea is used as a heuristic, since we cannot guarantee that a path will be in an optimal solution. However, in practice, this technique seems to work well.

Based on this idea, we choose branching variables as follows. Suppose that, at the root node, we branch on a variable x_{ef}. This means that, on the up-branch

node (i.e., the node in which $x_{ef} = 1$) and all of its descendants, edges e and f must belong to the same path. When making the next branching decisions, we try to expand this path, by branching on a variable associated with an angle that contains either e or f. For each descendant node, we continue this process until the path can no longer be extended, at which point we start a new path. In order to carry out this procedure, it is necessary to store, in each node of the search tree, the information concerning the paths induced by fixed variables. As we show in the next section, this strategy was quite effective in a number of instances.

2.3 Computational experiments

We now present a summary of our computational results. Experiments were run on an Intel Xeon CPU E5-2603, 1.60 GHz, with 32 GB RAM. Integer programs were solved with CPLEX 12.8 using traditional search with a single thread. The code was written in C++ and compiled with gcc 5.4.0. In all our runs, we imposed a time limit of 10 minutes.

In order to evaluate the performance of our ILP formulation, we generated graphs with orders chosen from the set $\{10, 15, 20, 25, 30, 35\}$. Instances were randomly generated by examining each pair of vertices u and v and adding the edge $\{u, v\}$ with a certain probability p, which is fixed for each instance and is chosen from the set $\{0.3, 0.5, 0.7, 0.9\}$. Thus, we have six different graph orders and four different edge insertion probabilities, yielding 24 configurations. For each configuration, we generate five graphs, obtaining a total of 120 instances.

We performed some preliminary experiments to determine which inequalities have the most impact on the performance of the ILP. The results showed that it is useful to add some cycle inequalities *a priori*, before starting the resolution of the ILP. We add all cycle inequalities associated with triangles of the input graph G. Additionally, we also add the star inequalities associated with all S_3 stars contained in G. Inequality (3) is also quite strong in practice, so we always add it, as well.

There exists an exponential number of cycle inequalities. Thus, they have to be separated during the execution of branch and cut. The separation in the case where the solution is integral is quite straightforward. Nevertheless, it is also possible to separate fractional cycle inequalities, using a procedure similar to one described by Grötschel et al. [17]. However, our experiments showed that running this separation procedure can be costly, and the resulting cutting planes do not help in the resolution of the problem. Therefore, we only run the separation procedure for integral solutions. Some of our results are summarized in Table 1. We compared the ILP with and without our branching strategy. Note how our strategy allows us to solve most problems with a drastic reduction on the number of nodes explored.

3 Checking Gallai's Conjecture for small graphs

In this section we detail how we used a computer to obtain the results in Theorem 1.2. One of the main challenges in this step is dealing with the huge amount of graphs for which it is required to check the conjecture, a total of 15.871.356.558

Table 1: Results of computational performance evaluation of Model 1. For each graph instance, we show its number of vertices (n), its density (*dens*), the number of nodes explored with CPLEX's branching strategy (*nod0*) and ours (*nod1*), primal values for both variants (*val0* and *val1*), best dual bound (same for both variants), and total time in seconds of each variant (*t0* and *t1*).

Instance	n	dens	nod0	nod1	val0	val1	dual	t0	t1
g20_30d	20	0.35	22443	181	5	5	5	273	0
g20_70c	20	0.72	27877	3647	14	9	9	600	56
g25_30a	25	0.34	49610	1386	8	6	6	600	7
g25_30b	25	0.33	3990	1798	7	7	7	34	15
g25_50e	25	0.50	24241	10527	13	8	8	600	73
g30_30d	30	0.30	16972	3839	9	8	8	600	45
g30_50a	30	0.48	19438	10642	16	12	12	600	218
g35_70a	35	0.72	2353	5149	326	20	14	600	600

connected non-isomorphic graphs (the storage space required to store these graphs in a compact representation is approximately 337 GB). To create all the graphs in the families considered by Theorem 1.2, we use *Nauty* [22], a program developed by B. D. McKay. Using this tool, we create all the possible non-isomorphic connected graphs with the following properties: (i) graphs with at most eleven vertices, (ii) bipartite graphs with at least twelve vertices and at most sixteen vertices, and (iii) $2k$-regular vertices with at least twelve vertices and at most fourteen vertices. Recall that Lovász [21] verified Conjecture 1.1 for graphs with at most one vertex with even degree. Therefore, there is no need to test Conjecture 1.1 for regular graphs with odd degree, and consequently no need to generate them. Table 2 summarizes the total amount of graphs divided by class and number of vertices.

To check Conjecture 1.1 for a graph G, we first check whether G can be a minimal counterexample (see Section 3.1). In the negative case, we simply dismiss G, and in the affirmative case, we proceed by searching for a *certificate* of Conjecture 1.1 for G, i.e., a path decomposition of G with at most $\lceil |V(G)|/2 \rceil$ paths. In the search for such a certificate, we apply two heuristics: first a BRKGA based heuristic (see Section 3.2), and then a heuristic that randomly fixes some paths (see Section 3.3). If both heuristics have failed to produce a certificate for G, then we solve Model 1 for G with the addition of inequality (5) with $t = \lceil |V(G)|/2 \rceil$. In this case, we say that we solve Model 1 with t as a *target*. This extra restriction allows us to stop the solver as soon as it finds a decomposition of size t.

$$|E(G)| - \sum_{\{e,f\} \in \mathcal{A}(G)} x_{ef} \geq t. \tag{5}$$

In what follows, we detail these steps separately.

3.1 Filters

In this section, we discuss the properties that we check in a given graph G to filter the graphs that cannot be minimal counterexamples. Suppose that we are verifying Conjecture 1.1 for a class of graph $\mathcal{G}$ and let G be a graph contained in such

Table 2: Number of non-isomorphic connected graphs. The *Total* column contains the total amount of graphs in that class. The *Filtered* column contains the total number of filtered graphs in that class (see Section 3.1), and the column "%" gives the percentage of such filtered graphs in relation to the total number of graphs. **Regular*:** the values for regular graphs with at least twelve vertices represent the number of connected non-isomorphic $2k$-regular graphs.

# vertices	All graphs			Bipartite graphs			Regular*
	Total	Filtered	%	Total	Filtered	%	Total
1	1	1	100,00%	1	1	100,00%	1
2	1	1	100,00%	1	1	100,00%	1
3	2	2	100,00%	1	1	100,00%	1
4	6	6	100,00%	3	3	100,00%	2
5	21	21	100,00%	5	5	100,00%	2
6	112	112	100,00%	17	17	100,00%	5
7	853	770	90,27%	44	44	100,00%	4
8	11.117	10.595	95,30%	182	182	100,00%	17
9	261.080	228.638	87,57%	730	730	100,00%	22
10	11.716.571	10.471.472	89,37%	4.032	4.029	99,93%	167
11	1.006.700.565	839.236.223	83,37%	25.598	25.549	99,81%	539
12				212.780	211.619	99,45%	9.489
13				2.241.730	2.228.157	99,39%	389.436
14				31.193.324	30.966.711	99,27%	25.157.396
15				575.252.112	570.119.622	99,11%	
16				14.218.209.962	14.050.400.053	98,82%	

class. We can skip the verification of Conjecture 1.1 for G if there exists a graph $G' \in \mathcal{G}$ such that $\mathrm{pn}(G) \leq \mathrm{pn}(G')$, $|V(G')| \leq |V(G)|$, and $|E(G')| < |E(G)|$. Note that if G' satisfies Conjecture 1.1, then this implies that G also satisfies it, since $\mathrm{pn}(G) \leq \mathrm{pn}(G') \leq \lceil |V(G')|/2 \rceil \leq \lceil |V(G)|/2 \rceil$. The condition $G' \in \mathcal{G}$ is necessary to guarantee that G' will be tested at some point during the verification process of class $\mathcal{G}$, and condition $|E(G')| < |E(G)|$ guarantees that we will not dismiss G in favor of G' and G' in favor of G. In what follows, we describe these filters.

Liftings. Suppose that G contains a vertex v of degree 2 such that its neighbors, say x and y, are non-adjacent. A *lifting* on v is the operation of removing v and joining its neighbors, which yields the graph $G' = G - v + xy$. Let $\mathcal{D}'$ be a minimum path decomposition of G', let $P' \in \mathcal{D}'$ be the path that contains xy, and let P be the path obtained from P' by replacing the edge xy by the subpath xvy. Note that $\mathcal{D} = \mathcal{D}' - P' + P$ is a path decomposition of G such $\mathrm{pn}(G) \leq |\mathcal{D}| = |\mathcal{D}'| = \mathrm{pn}(G')$. Therefore, we can dismiss all graphs that contain a vertex with degree 2 whose neighbors are non-adjacent.

Maximum degree at most 5. Bonamy and Perret [1] verified Conjecture 1.1 for graphs with maximum degree at most 5, therefore we can dismiss all such graphs.

Even subgraph. Lovász [21], Pyber [24], and Fan [10] studied the so-called *even subgraph* of a graph G. The even subgraph G_E of a graph G is the subgraph of G induced by its vertices with even degree. Alternatively, G_E is the graph obtained from G by removing its vertices with odd degree. Conjecture 1.1 was verified by Lovász [21] in the case G_E contains at most one vertex; by Pyber [24] in the case G_E is a forest; and by Fan [10] in the case each block of G_E has maximum degree at

most 3 and no triangles. Clearly, G_E can be constructed in linear time. Moreover, whether G_E is a forest can also be decided in linear time. Fan's results, on the other hand, can be used in a more efficient manner. The following result formalizes the even subgraph filter. In what follows, given a vertex $v \in V(G)$, we denote by $\delta_E(v)$ the set of edges incident to v in G_E, by *E-degree*, we mean the degree in G_E, and by *E-neighbor*, we mean a neighbor in G_E.

Lemma 3.1 *Let G be a graph,*

(i) If G_E is a forest, then $\mathrm{pn}(G) \leq \lfloor |V(G)|/2 \rfloor$;

(ii) If G_E contains a vertex v with odd E-degree and at most one E-neighbor with E-degree greater than 3, then $\mathrm{pn}(G) \leq \mathrm{pn}(G - \delta_E(v))$;

(iii) If G_E contains two adjacent vertices v and u having no common E-neighbor and such that v has even E-degree and at most one E-neighbor with E-degree greater than 3, and u has E-degree 2, then $\mathrm{pn}(G) \leq \mathrm{pn}(G - \delta_E(v) - \delta_E(u) + uv)$.

Lemma 3.1(i) guarantees that a graph satisfies Conjecture 1.1. Lemmas 3.1(ii) and (iii), on the other hand, test whether the given graph is a minimum counterexample. In each of these cases, we construct a subgraph G' of G such that $\mathrm{pn}(G) \leq \mathrm{pn}(G')$, and hence we can dismiss G. Clearly, all these properties can be tested in linear time.

Table 2 shows, for each of the classes of graphs considered in this paper, the number and percentage of the those that were filtered by the filters presented in this section. We remark that, when checking Conjecture 1.1 for bipartite graphs with at least thirteen vertices, we disabled the lifting filter. The reason is that this could lead to a non-bipartite graph with twelve vertices that would never be checked, since we generated only bipartite graphs and regular graphs with more than eleven vertices. This is also the reason why Table 2 contains no column showing the number of regular graphs filtered, as we had to disable all the filters for them.

3.2 Biased random-key genetic algorithms

Biased Random-Key Genetic Algorithm (BRKGA) [14,15,16] is a search metaheuristic for combinatorial optimization problems. As the name indicates, it is a genetic algorithm and, as such, relies on an evolutionary process to obtain high-quality solutions. Following the commonly used terminology, a solution to a problem of interest is called a *chromosome*. Its objective function value is referred to as its *fitness* and a chromosome associated with its fitness value is said to be an *individual*. The algorithm operates on a set of individuals called the *population* and each iteration corresponds to a *generation*. The population is subject to an evolutionary process inspired by natural selection, in which the best fit individuals have a higher chance of producing offspring, whereas worst-fit individuals tend to be replaced.

A key aspect of BRKGA is that it adopts a standardized problem-independent encoding for the solutions. Each chromosome consists of a fixed number of *random keys* (or *alleles*), which are real numbers over the interval $[0, 1)$. Therefore, all steps of the evolution process are also problem-independent. The connection with each

specific problem is achieved by providing a deterministic algorithm called *decoder*, which computes the fitness of each chromosome. This considerably decreases coding effort and makes BRKGA especially appealing.

Now, we describe the decoder that we propose. Much like in our ILP model, solutions given by our BRKGA heuristic consist of sets of angles. Given an input graph G, each chromosome has $|\mathcal{A}(G)|$ alleles, each one associated with an angle of G. We construct a path decomposition by iteratively adding angles to an initially empty solution. Angles are processed in a greedy fashion, in non-increasing order of their associated allele values. An angle a can be added to a partial solution S if (i) S has no angles that share an edge with a and that are supported by the same vertex as a; and (ii) the inclusion of a does not introduce cycles to any elements of the decomposition induced by S. Note that conditions (i) and (ii) are analogous to constraints (1b) and (1c) of our ILP model, respectively.

We implemented our BRKGA heuristic using `libbrkga` [25], a library developed by Silva, Resende, and Pardalos. The evolutionary process is controlled by the following parameters: (i) fraction of the population consisting of elite individuals; (ii) fraction of the population to be replaced by mutants; and (iii) probability that an offspring inherits an allele from its elite parent. We set these parameters to 10%, 20% and 70%, respectively (see [25] for more details on them). We also observed that the heuristic is often able to find a certificate for a given graph G with a very small population and few generations. Therefore, we employed the following strategy. We start with a population of size 2 and evolve it for four generations. If no certificate is found, we restart the BRKGA with a new population containing twice as many individuals and run another four generations. This process continues until a certificate is found or the population size grows above 32.

3.3 Random Path Removal Heuristic

In this section, we describe the heuristic used when the BRKGA has failed to produce a certificate for Conjecture 1.1. The main difficulty encountered by the BRKGA heuristic is to deal with dense graphs, due to the high number of variables and their symmetry. To overcome this problem, we randomly choose edge-disjoint paths to add to the decomposition, which results in a graph with fewer variables and, hopefully, less symmetries.

Our heuristic searches for a certificate for a graph G_0 in the following way. For $i = 1, \ldots, k$, let $G_i = G_{i-1} - E(P_i)$, where P_i is a randomly chosen maximal path in G_{i-1}. Additionally, let $D' = \{P_1, \ldots, P_k\}$, let n_{odd} be the number of vertices with odd degree in G_k, and let

$$\varphi = \left\lceil \max \left\{ \frac{\Delta(G_k)}{2}, \frac{n_{\text{odd}}}{2}, \frac{|E(G_k)|}{(|V(G_k)| - 1)} \right\} \right\rceil. \tag{6}$$

Note that if $\varphi > \lceil |V(G)|/2 \rceil - k$, then there is no decomposition of G_k that, when joining to D', composes a certificate for G_0. We use this condition to eliminate bad candidates to a certificate of G_0. If G_k passes the test condition, then we solve

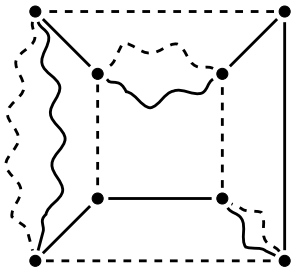

Figure 1. A planar graph composed of a path and a cycle with path number 3 in a graph with any girth. We use a snaked line to denote a path with possibly several internal vertices.

Model 1 for G_k with target $\lceil |V(G)|/2 \rceil - k$, and with a stop condition of one second, i.e., the solver has only one second to solve the model. If the solver is able to solve the model, returning a decomposition $\mathcal{D}_k$, then $\mathcal{D}_k \cup \mathcal{D}'$ is a certificate for G. Otherwise, it restarts by choosing another collection D' of paths to remove from G. In our tests, we repeated this procedure until a certificate for G was found or 100 attempts failed.

4 Concluding remarks and future works

In this paper, we presented an ILP model for computing the path number of a graph G and discussed how to use it, together with heuristics and known theoretical results, for verifying Conjecture 1.1 for a large collection of graphs. Applying the ideas discussed in this work, we were able to verify all the 11.716.571 non-isomorphic connected graphs with ten vertices within 35 minutes, using a single thread in a computer equipped with a processor *Intel® Xeon® E5620 2.40GHz*. This means that, on average, we were able to check Conjecture 1.1 for 5500 graphs per second. This performance allowed us to verify Conjecture 1.1 for all the 15.871.356.558 graphs required to confirm Theorem 1.2 in less than a week using 10 threads in the same computer.

Another outcome provided by Model 1 has come when studying triangle-free planar graphs. By using a result of Lovász [21], one can reduce the problem of checking Gallai's Conjecture for this class to the problem of proving that every triangle-free planar graph consisting of a cycle and a path has path number 2. By generating small graphs of this type and evaluating them using Model 1, we found counterexamples, concluding that this strategy cannot work for this class of graphs. In fact, this strategy does not work for planar graphs with any girth (see Figure 1), and Conjecture 1.1 was verified for triangle-free planar graphs using a different approach [5].

In future works, we plan to verify Conjecture 1.1 for all the 164.059.830.476 graphs with twelve vertices (a number of graphs approximately 163 times higher than the number of graphs with eleven vertices). Moreover, we plan to compute the path number of these graphs. In another direction, we believe that Model 1 can be used to derive an approximation algorithm for the problem of computing the path number of a graph.

A full version of this work will be submitted for publication with a more detailed and individual analysis of each optimization constraint. Lastly, both our instance benchmark and our code will be available online, so that it may become a tool for other researchers in this area.

References

[1] Bonamy, M. and T. J. Perrett, *Gallai's path decomposition conjecture for graphs of small maximum degree*, Discrete Math. **342** (2019), pp. 1293–1299.

[2] Bondy, A., *Beautiful conjectures in graph theory*, European J. Combin. **37** (2014), pp. 4–23.

[3] Bondy, J. A. and U. S. R. Murty, "Graph theory," Graduate Texts in Mathematics **244**, Springer, New York, 2008.

[4] Botler, F. and A. Jiménez, *On path decompositions of* $2k$*-regular graphs*, Discrete Math. **340** (2017), pp. 1405–1411.

[5] Botler, F., A. Jiménez and M. Sambinelli, *Gallai's path decomposition conjecture for triangle-free planar graphs*, Discrete Math. **342** (2019), pp. 1403 – 1414.

[6] Botler, F., M. Sambinelli, R. S. Coelho and O. Lee, *On Gallai's and Hajós' conjectures for graphs with treewidth at most* 3, ArXiv e-prints (2017), submitted.

[7] Constantinou, C. K. and G. Ellinas, *Minimal path decomposition of complete bipartite graphs*, J. Comb. Optim. **35** (2018), pp. 684–702.

[8] Dean, N. and M. Kouider, *Gallai's conjecture for disconnected graphs*, Discrete Math. **213** (2000), pp. 43–54, selected topics in discrete mathematics (Warsaw, 1996).

[9] Donald, A., *An upper bound for the path number of a graph*, J. Graph Theory **4** (1980), pp. 189–201.

[10] Fan, G., *Path decompositions and Gallai's conjecture*, J. Combin. Theory Ser. B **93** (2005), pp. 117–125.

[11] Favaron, O. and M. Kouider, *Path partitions and cycle partitions of Eulerian graphs of maximum degree* 4, Studia Sci. Math. Hungar. **23** (1988), pp. 237–244.

[12] Geng, X., M. Fang and D. Li, *Gallai's conjecture for outerplanar graphs*, Journal of Interdisciplinary Mathematics **18** (2015), pp. 593–598.

[13] Glock, S., D. Kühn and D. Osthus, *Optimal path and cycle decompositions of dense quasirandom graphs*, Journal of Combinatorial Theory, Series B **118** (2016), pp. 88–108.

[14] Gonçalves, J. F. and M. G. C. Resende, *Biased random-key genetic algorithms for combinatorial optimization*, Journal of Heuristics **17** (2011), pp. 487–525.

[15] Gonçalves, J. F. and M. G. C. Resende, "Random-Key Genetic Algorithms," Springer International Publishing, Cham, 2018 pp. 703–715.

[16] Gonçalves, J. F. and M. G. Resende, *An evolutionary algorithm for manufacturing cell formation*, Computers & Industrial Engineering **47** (2004), pp. 247–273.

[17] Grötschel, M., M. Jünger and G. Reinelt, *On the acyclic subgraph polytope*, Mathematical Programming **33** (1985), pp. 28–42.

[18] Harding, P. and S. McGuinness, *Gallai's conjecture for graphs of girth at least four*, Journal of Graph Theory **75** (2014), pp. 256–274.

[19] Heinrich, I., M. V. Natale and M. Streicher, *Hajós' cycle conjecture for small graphs*, ArXiv e-prints (2017).

[20] Jiménez, A. and Y. Wakabayashi, *On path-cycle decompositions of triangle-free graphs*, Discrete Math. Theor. Comput. Sci. **19** (2017), pp. Paper No. 7, 21.

[21] Lovász, L., *On covering of graphs*, in: *Theory of Graphs (Proc. Colloq., Tihany, 1966)*, Academic Press, New York, 1968 pp. 231–236.

[22] McKay, B. D. et al., *Practical graph isomorphism* (1981).

[23] Péroche, B., *NP-completeness of some problems of partitioning and covering in graphs*, Discrete Appl. Math. **8** (1984), pp. 195–208.

[24] Pyber, L., *Covering the edges of a connected graph by paths*, J. Combin. Theory Ser. B **66** (1996), pp. 152–159.

[25] Silva, R. M., M. G. Resende and P. M. Pardalos, *A python/c++ library for bound-constrained global optimization using a biased random-key genetic algorithm*, Journal of Combinatorial Optimization **30** (2015), pp. 710–728.