Isomorphism for graphs of bounded feedback vertex set number

Stefan Kratsch and Pascal Schweitzer
Max-Planck-Institute for Computer Science
Campus E1 4, 66123 Saarbrücken, Germany
{skratsch,pascal}@mpi-inf.mpg.de

December 10, 2009

Abstract

This paper presents an $O(n^2)$ algorithm for deciding isomorphism of graphs that have bounded feedback vertex set number. This number is defined as the minimum number of vertex deletions required to obtain a forest. Our result implies that Graph Isomorphism is fixed-parameter tractable with respect to the feedback vertex set number. Central to the algorithm is a new technique consisting of an application of reduction rules that produce an isomorphism-invariant outcome, interleaved with the creation of increasingly large partial isomorphisms.

1 Introduction

The Graph Isomorphism problem is among the few problems in $\mathcal{NP}$ for which the complexity is still unknown: Up to now, neither an $\mathcal{NP}$-hardness proof nor an algorithm with provably polynomial running time has appeared. Given two finite graphs G_1 and G_2, the Graph Isomorphism problem (GI) asks whether these graphs are structurally equivalent, i.e., whether there exists a bijection from $V(G_1)$, the vertices of G_1, to $V(G_2)$, the vertices of G_2, that preserves the adjacency relationship. Being one of the open problems from Garey and Johnson's list of problems with yet unsettled complexity status [17], the Graph Isomorphism problem has been studied extensively throughout the last three decades. During that time, a subexponential-time algorithm for the general problem has been developed by Babai [2]. His algorithm uses a degree reduction method by Zemlyachenko (see [2]) as well as Luks' polynomial-time algorithm for graphs of bounded degree [21]. Schöning's lowness proof [26] showed that Graph Isomorphism is not $\mathcal{NP}$-hard, unless the polynomial hierarchy collapses.

Research on Graph Isomorphism for restricted graph classes has led to a number of polynomial-time algorithms as well as hardness results. Let us review the known results for classes defined by bounded values of some graph parameter, e.g., graphs of degree bounded by k, from a parameterized point of view. Depending on the parameter GI becomes polynomial-time solvable or it remains GI-complete (i.e., polynomial-time equivalent to GI) even when the parameter is bounded by a constant. The latter is known for bounded chromatic number and bounded chordal deletion number (i.e., number of vertex deletions needed to obtain a chordal graph), since Graph Isomorphism is GI-complete for the class of bipartite graphs and the class of chordal graphs (see [30]).

The polynomial results can be further split into runtimes of the form $O(f(k)n^c)$ and $O(n^{f(k)})$; both are polynomial for bounded k but for the latter the degree of the polynomial grows with k.

Parameter	Comp. of the parameter		Upper bound for GI	
Chromatic number	3-col. is $\mathcal{NP}$-hard	[17]	GI-hard for $\chi(G) = 2$	
Chordal deletion number	$O(f(k)n^c)$	[22]	GI-hard for $\mathrm{cvd}(G) = 0$	
Max degree	$O(m)$		$O(n^{ck})$	[4]
Genus	$O(f(k)m)$	[19]	$O(n^{ck})$	[13, 23]
Treewidth	$O(f(k)n)$	[6]	$O(n^{k+4.5})$	[5]
Rooted tree distance width	$O(kn^2)$	[31]	$O(f(k)n^3)$	[31]
$\mathcal{H}$-free deletion number	$O(d^k n^d)$	[7]	FPT if $(*)$	[Section 2]
Feedback vertex set number	$O(5^k kn^2)$	[8]	$O((2k + 4k \log k)^k kn^2)$	[Section 3]

Table 1: For various graph parameters the table shows upper bounds on the running time required to compute the parameter as well as running times of the Graph Isomorphism problem. $(*)$ The fixed-parameter tractability holds if the colored $\mathcal{H}$-free isomorphism problem can be solved in polynomial time.

Parameterized complexity (see [9]) studies these function classes in a multivariate analysis of algorithms, motivated by the much better scalability of $O(f(k)n^c)$ algorithms, so-called *fixed-parameter tractable* algorithms. In the case of Graph Isomorphism for a large number of parameters only $O(n^{f(k)})$ algorithms are known. Such algorithms exist for the parameters degree [21], eigenvalue multiplicity [3], color class size [15], and treewidth [5]. Furthermore, this running time has been shown for the parameter genus [13, 23] (extending polynomial-time algorithms for planar graphs [18, 27]) and, more general, for the size of an excluded minor [24]. Algorithms of runtime $O(f(k)n^c)$ are known for the parameters color multiplicity [16], eigenvalue multiplicity [10], rooted tree distance width [31]. For chordal graphs, there is an fpt-algorithm with respect to the size of the simplicial components [29]. The fixed-parameter tractable algorithm for the parameter color multiplicity has recently been extended to hypergraphs [1]. Table 1 summarizes some of these results for parameterized Graph Isomorphism as well as the complexity of computing the parameters.

We develop an $O(f(k)n^c)$ algorithm for Graph Isomorphism parameterized by the feedback vertex set number. The *feedback vertex set number* of a graph G, denoted by $\mathrm{fvs}(G)$, is the size of a smallest subset of vertices S, whose removal leads to a graph that does not contain cycles, i.e., for which $G - S$, the graph induced by the set of vertices $V(G) \setminus S$, is a forest. Our result, a fixed-parameter tractable algorithm, has a running time of $O(f(k)n^2)$, i.e., it runs in $O(n^2)$ for graphs of bounded feedback vertex set number.

For a selection of graph parameters, Figure 1 shows the partial order given by the relation stating that a parameter k' is larger than another parameter k, if k can be bounded by a function g of k'. From this it is immediate that if a problem is fixed-parameter tractable (FPT) with respect to some parameter then it is also FPT with respect to any larger (in Figure 1 higher) parameter: time $O(f(k)n^c)$ with respect to k implies time $O(f(g(k'))n^c)$ with respect to k' (likewise for runtimes of the form $O(n^{f(k)})$). The feedback vertex set number, which has been extensively studied in various contexts [8, 11, 14, 17, 25, 28], lies above other interesting parameters: As mentioned GI remains hard on graphs of bounded chromatic number, while being polynomially solvable for bounded treewidth. As the rooted tree distance width the feedback vertex set number is a measure

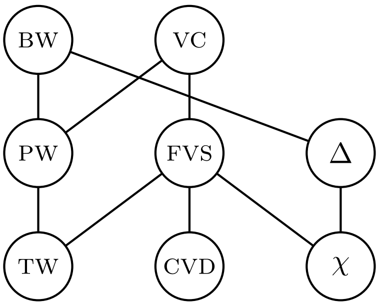

Figure 1: For various graph parameters, the figure depicts the partial order given by the relation that defines a parameter to be lower than another parameter, if the former can be bounded by a function of the latter. If a problem is FPT with respect to some parameter then it is also FPT with respect to all higher parameters. The parameters are: bandwidth (BW), pathwidth (PW), treewidth (TW), size of a minimum vertex cover (VC), size of a minimum feedback vertex set (FVS), vertex deletion distance from chordal graphs (CVD), maximum degree (Δ), and chromatic number (χ).

for how far a graph is from being a forest. However, these two parameters are incomparable, i.e., neither is bounded by a function of the other.

Our contribution is based on two new techniques: The first makes use of the interplay between deletion sets and small forbidden structures. This is illustrated in Section 2 on the simplified situation where the parameter is the vertex deletion distance to a class of graphs that is characterized by finitely many forbidden induced subgraphs. When we consider the feedback vertex set number in Section 3, the forbidden substructures are cycles, which may be of arbitrary length. The second technique addresses this obstacle by using reduction rules that guarantee short cycles. For the choice of these rules, however, it is crucial that they are compatible with isomorphisms.

2 $\mathcal{H}$-free deletion number

In this section, illustrating the usefulness of deletion sets in the context of GRAPH ISOMORPHISM, we briefly consider the parameter $\mathcal{H}$-free deletion number. For a class $\mathcal{C}$ of graphs we say that a graph G has *vertex deletion distance at most k from $\mathcal{C}$* if there is a *deletion set S* of at most k vertices, for which $G - S \in \mathcal{C}$, i.e., by deleting a most k vertices we obtain a graph in $\mathcal{C}$.

Definition 1. A class $\mathcal{C}$ of graphs is *characterized by finitely many forbidden induced subgraphs*, if there is a finite set of graphs $\mathcal{H} = \{H_1, \ldots, H_\ell\}$, such that a graph G is in $\mathcal{C}$ if and only if G does not contain H_i as an induced subgraph for any $i \in \{1, \ldots, \ell\}$. The class $\mathcal{C}$ is called the class of $\mathcal{H}$-free graphs.

It is known that computing the $\mathcal{H}$-free deletion number k and a corresponding set S of vertices to be removed is FPT with respect to k: There is an algorithm with runtime $O(d^k n^d)$ where d is the number of vertices of the largest forbidden induced subgraph. This follows from a more general result for graph modification problems due to Cai [7]. In fact the special case of vertex deletion can be solved by identifying all sets of at most d vertices that induce a forbidden subgraph and computing a minimum hitting set; using a current d-HITTING SET algorithm [12] this gives a slightly better exponential dependence on k of c_d^k where $c_d \leq d$ and c_d tends to $d - 1$.

For an FPT-algorithm that solves Graph Isomorphism parameterized by the $\mathcal{H}$-free deletion number, we require a method of consistently removing vertices from the graph: Let G be a colored graph, c a vertex coloring of G, and v a vertex of G. We define $G \triangleright v$ to be the colored graph on the vertex set $V(G) \setminus \{v\}$ with the coloring given by $(\chi(v, v'), c(v'))$ for all $v' \in V(G) \setminus \{v\}$, where the edge characteristic function $\chi(v, v')$ is 1 if v and v' are adjacent and 0 otherwise. Intuitively, the new coloring encodes at the same time whether in the original graph a vertex is adjacent to v as well as its previous color. In particular, we get the following observation: Suppose that G_1 and G_2 are colored graphs and that v_1 and v_2 are equally colored vertices of G_1 and G_2 respectively, then there is an isomorphism ϕ with $\phi(v_1) = v_2$ if and only if $G_1 \triangleright v_1$ and $G_2 \triangleright v_2$ are isomorphic as colored graphs (where isomorphisms must respect colors).

Algorithm 1 $\mathrm{IsomorphismIND}_{\mathcal{C}}$

Input: (G_1, G_2, k): A parameter k and two colored graphs G_1, G_2 with distance of at most k to a fixed class $\mathcal{C}$ that is characterized by a finite set of forbidden induced subgraphs.

Output: An isomorphism ϕ of G_1 and G_2 or report **false** if no such isomorphism exists.

```
if exactly one of $G_1$ and $G_2$ contains a forbidden subgraph then
  return false
end if
if $G_1$ and $G_2$ contain no forbidden subgraphs then
  use an algorithm for colored graph isomorphism for the class $\mathcal{C}$ on $G_1$ and $G_2$
  return an isomorphism or false if none exists
end if
if $k = 0$ then
  return false
end if
find a forbidden induced subgraph $H$ in $G_2$
find a set $S$ of at most $k$ vertices such that $G_1 - S \in \mathcal{C}$
for all $(v_1, v_2) \in S \times V(H)$ do
  if $v_1$ and $v_2$ are colored with the same color then
    result ← $\mathrm{IsomorphismIND}_{\mathcal{C}}(G_1 \triangleright v_1, G_2 \triangleright v_2, k - 1)$
    if result $\neq$ false then
      return result
    end if
  end if
end for
return false
```

Theorem 1. *Let the graph class $\mathcal{C}$ be characterized by the forbidden induced subgraphs $H_1, \ldots, H_\ell$. If the colored graph isomorphism problem for graphs from $\mathcal{C}$ is in $\mathcal{P}$, then the colored graph isomorphism problem, parameterized by the vertex deletion distance from $\mathcal{C}$, is fixed-parameter tractable.*

Proof. W.l.o.g., both input graphs G_1 and G_2 have distance of at most k from $\mathcal{C}$. Algorithm 1 repeatedly generates a set of candidate pairs of vertices $P = S \times V(H) \subseteq V(G_1) \times V(G_2)$, where S is a minimum deletion set of G_1 and $V(H)$ is the vertex set of a forbidden induced subgraph H in G_2. For any isomorphism ϕ, this ensures that P contains a pair (v_1, v_2) with $\phi(v_1) = v_2$. Indeed, the image $\phi(S)$ is a deletion set of G_2, thereby also intersecting H. The algorithm then makes a recursive call for each choice of $(v_1, v_2) \in P$, removing the vertices v_1 and v_2 from the graphs and correctly coloring the remaining vertices. In the base case, when $G_1, G_2 \in \mathcal{C}$, isomorphism is decided using the polynomial-time algorithm for the class $\mathcal{C}$.

Observe that P has size at most dk where d is the size of the largest graph in $\{H_1, \ldots, H_\ell\}$. Thus there are $O((dk)^k)$ recursive calls, since k decreases with each call and $k = 0$ terminates a branch. Together with the polynomial-time algorithm for the base case and the FPT-algorithm for distance from $\mathcal{C}$ [7] this gives an $O(f(k)n^c)$ runtime, proving fixed-parameter tractability. ☐

As an example of a class that satisfies the assumption of Theorem 1, we mention the graphs of fixed bounded degree $d \in \mathbb{N}$, which have a characterization by finitely many forbidden subgraphs, and for which Luks [21] describes a polynomial-time algorithm that also solves GRAPH ISOMORPHISM for colored graphs. Note however, that in this case the problem is not parameterized by the degree, but by the vertex deletion distance to a graph with maximum degree of at most d.

3 Feedback vertex set number

In this section we consider the GRAPH ISOMORPHISM problem parameterized by the feedback vertex set number. Similarly to Section 2 we compute a set that intersects all forbidden structures of the first graph (in our case a feedback vertex set). The image of that set under any isomorphism must intersect every forbidden structure of the second graph (i.e., it must intersect every cycle). To efficiently use this fact, we choose a shortest cycle in the second graph. However, since in general shortest cycles may be of logarithmic size, we perform a sequence of reductions to shorten the cycles.

The reduction rules delete all vertices of degree at most one as well as those in a specified set S, and contract vertices of degree two; these are standard reductions for computing feedback vertex sets. Additionally, there is a new rule resulting in the deletion of all components containing at most one cycle. This rule allows us to prove the crucial fact, that exhaustive reduction of graphs behaves well with respect to isomorphism. In order to make this precise, we first show that the result of exhaustively applying the reduction rules is independent of the order in which they are applied.

Lemma 1. *Let G be a graph and let S be a set. Exhaustive application of the following reduction rules in any order has a well-defined result $R_S(G)$, which is a specific graph on a subset of $V(G)$.*

1. *Delete a vertex of degree at most one.*
2. *Delete a vertex in a connected component containing at most one cycle.*
3. *Delete a vertex that is contained in S.*
4. *Contract a vertex of degree two that is not contained in S, i.e., replace the vertex by an edge between its former neighbors; this may create multi-edges and loops.*

Proof. For any graph G and any set S let $L_S(G)$ denote the maximum number of reduction steps that can be applied to G (using S for Rule 3).

We assume for contradiction that there is a counterexample consisting of a graph G and a set S with minimum value of $L_S(G)$. Let R_1 and R_2 be two maximal sequences of reduction steps for G that yield different results. For $i \in \{1, 2\}$ let v_i be the first vertex reduced by R_i and let G_i

be the result of that first step. Observe that $L_S(G_1) < L_S(G)$ and $L_S(G_2) < L_S(G)$, implying that $R_S(G_1)$ and $R_S(G_2)$ are well-defined, by our choice of G.

It suffices for us to show that we can reduce v_2 in G_1 and v_1 in G_2 such that we obtain the same graph G' on the same subset of $V(G)$: Indeed since any further exhaustive reduction has the same outcome, this implies $R_S(G_1) = R_S(G') = R_S(G_2)$ since the result of any maximal sequence of reductions on either G_1 and G_2 is well-defined, and yields the desired contradiction.

The deletion rules (Rules 1–3) are such that a vertex that may be deleted in a graph G may also be deleted in any subgraph of G. Therefore, if *both vertices are deleted from* G, then v_2 can be deleted from G_1 and v_1 can be deleted from G_2; we obtain $G' = G - \{v_1, v_2\}$.

Otherwise *w.l.o.g.* v_1 *is contracted in* G; there are three cases:

1. If v_2 *is not adjacent to* v_1 then the reductions are independent and reducing v_1 in G_2 and v_2 in G_1 yields the same graph G' on the vertex set $V(G) \setminus \{v_1, v_2\}$.

2. If v_2 *is contracted and adjacent to* v_1, then there is a path (u, v_1, v_2, w) and contracting v_1 and v_2 in any order is equivalent to replacing the path by an edge $\{u, w\}$, reducing both graphs to the same graph G' on the vertex set $V(G) \setminus \{v_1, v_2\}$.

3. If v_2 *is deleted and adjacent to* v_1, then the degree of v_2 in G_1 is the same as in G, therefore it can still be deleted. In G_2 the vertex v_1 has degree at most 1, implying that it can be deleted by Rule 1. Both reductions lead to the same graph $G' = G - \{v_1, v_2\}$. ☐

Let us observe that $R_S(G)$ has minimum degree at least three and that $\mathrm{fvs}(R_S(G)) \leq \mathrm{fvs}(G)$ for any graph G and any set S. Concerning the latter, it suffices to observe that vertex deletions do not increase the feedback vertex set number, and that any degree-2-vertex of a feedback vertex set may be replaced by either neighbor while preserving the property of being a feedback vertex set. We denote by $R(G) := R_{\{\}}(G)$ the special case that S is the empty set. Since the result is independent of the order in which the rules are applied, the vertices from the set S may be removed first, i.e., $R_S(G) = R(G - S)$.

As a corollary of Lemma 1 we conclude that the reduction R maintains isomorphisms.

Corollary 1. *Let ϕ be an isomorphism of two graphs G_1 and G_2 and let $S \subseteq V(G_1)$. Then $R_S(G_1)$ and $R_{\phi(S)}(G_2)$ are isomorphic and ϕ restricted to $V(R_S(G_1))$ is an isomorphism from $R_S(G_1)$ to $R_{\phi(S)}(G_2)$.*

The reduction rule that allows vertex deletion in unicyclic components, which is necessary to obtain Corollary 1, has the effect that the set S does not need to be a feedback vertex set, in order for the reduced graph $R_S(G)$ to become empty. As a consequence, removal of such a set S does not necessarily leave a forest, but a graph that may contain a cycle in every component.

Definition 2. An *OC graph* is a graph in which every component contains at most one cycle.

The OC graphs are precisely the graphs which are reduced to the empty graph by repeated application of the reduction rules:

Lemma 2. *A graph G is an OC graph if and only if $R(G)$ is the empty graph.*

Proof. To show that for an arbitrary OC graph G the reduced graph $R(G)$ is the empty graph we assume, w.l.o.g., that the graph is connected. Thus G contains at most one cycle. We claim that

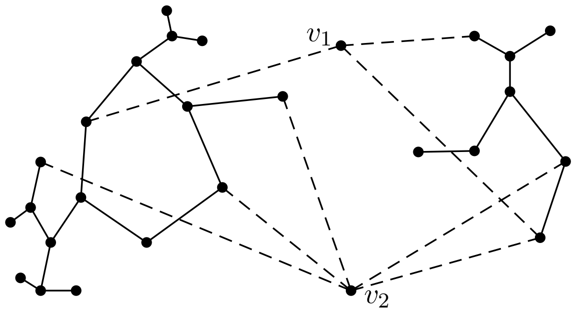

Figure 2: An OC+2 Graph with distinguished vertices (v_1, v_2). After removal of v_1 and v_2 the graph consists of two components: a tree and a unicyclic graph.

after repeatedly removing all vertices of degree at most 1, the graph is empty, or is a cycle: Indeed, suppose v is a vertex in $V(G)$ not contained in a cycle, and v is not removed by the reductions, then in the reduced graph v has degree at least 2, and the longest path through v in the reduced graph ends on one side with a vertex of degree 1, a contradiction. Finally by induction every cycle reduces to the empty graph.

Conversely, to show that a graph which reduces to the empty graph is an OC graph, it suffices to show that any graph which contains two cycles in one component does not reduce to the empty graph. The minimal connected graphs that contain two cycles are the dumbells (i.e., two cycles joined by a path) and the Theta graphs (i.e., two vertices connected by three vertex disjoint paths). By induction they do not reduce to the empty graph, and the lemma follows. ☐

For reduced graphs, we can use a nice structural result by Raman, Saurabh and Subramanian [25], stating that graphs of minimum degree at least three must have a large feedback vertex set number or a cycle of length at most six. Thus, in contrast to the general bound of $\log n$ on the girth of a graph, there are few choices for the image of any feedback vertex under an isomorphism between two reduced graphs.

Theorem 2 ([25]). *Let G be a graph on n vertices with minimum degree at least three and of feedback vertex set number at most k. If $n > 2k^2$ then G has a cycle of length at most six.*

The algorithm that we present later branches on possible partial isomorphisms; using Theorem 2 and our reductions the number of choices is reasonably small. On termination there are pairs of vertices that the isomorphism shall respect and removal of those vertices followed by reduction yields two empty graphs. Hence, deletion of the vertices yields two OC graphs. This leaves us with the task of deciding isomorphism for OC graphs, with the restriction that adjacencies with the deleted vertices must be correct. For that purpose we first define OC+k graphs and corresponding isomorphisms.

Definition 3. A graph with *at most one cycle per component plus k distinguished vertices* (OC+k graph) consists of a graph and a k-tuple of its vertices with the property that deletion of those distinguished vertices yields an OC graph.

An *isomorphism of two OC+k graphs* is an ordinary isomorphism that maps the k distinguished vertices of one graph to those of the other respecting the order. If the graph is (vertex) colored, then as usual the isomorphism has to respect these colors.

The restriction on the mapping of the k-tuple of vertices allows an efficient decision of isomorphism for these graphs, mainly requiring an isomorphism test of colored OC graphs.

Theorem 3. Graph Isomorphism *for colored OC+k graphs can be solved in* $O(n^2)$ *time.*

Proof. We first reduce the problem to colored OC graphs. For this it suffices to reduce the problem for $k > 1$ to the isomorphism problem of OC+$(k-1)$ graphs in $O(n)$ time. Let G and G' be two given colored OC+k graphs with last distinguished vertex v_k and v'_k respectively. If v_k and v'_k are not equally colored, then the graphs are not isomorphic. Otherwise, as argued in the previous section, the graphs $G_1 \triangleright v_k$ and $G_2 \triangleright v'_k$ are isomorphic, if and only if G_1 and G_2 are isomorphic. The recoloring and the vertex deletion of the reduction require $O(n)$ time.

We are left with determining the isomorphism of colored OC graphs G_1 and G_2. We assign every vertex v, neighboring a leaf and contained in a component with at least 3 vertices, a color that depends on the multiset of colors of leaf neighbors of the vertex v. We then delete all leaves in these components. Again, the obtained graphs are isomorphic if and only if they were isomorphic prior to the reduction. After this, we rename (in both graphs consistently) the new colors with unused integers in $\{1, \ldots, n\}$ by sorting. By repeated application we obtain graphs in which every component is a cycle or contains at most 2 vertices. This step can be performed in an amortized time of $O(n \log(n))$, charging the sorting to the removed leaves.

Counting for each isomorphism type the number of components with at most two vertices, it suffices now to determine the isomorphism of disjoint unions of colored cycles. There are at most n such cycles in an OC graph. We solve this task using a string matching algorithm: A colored cycle $\langle c_1, c_2, \ldots, c_n \rangle$ is isomorphic to $\langle c'_1, c'_2, \ldots, c'_{n'} \rangle$ if and only if $n = n'$ and the string $c'_1 c'_2 \ldots c'_n$ or its inverse $c'_n c'_{n-1} \ldots c'_1$ is contained as a consecutive substring in the string $c_1 c_2 \ldots c_n c_1 c_2 \ldots c_n$. We repeatedly search two color-isomorphic cycles from each graph and remove them. By employing a linear time string matching algorithm, like the Knuth-Morris-Pratt algorithm [20], we obtain a total running time of $O(n^2)$. ☐

Knowing how to efficiently decide isomorphism of OC+k graphs, we work towards an algorithm that creates sets of pairs of distinguished vertices, such that each isomorphism of the given graphs must respect one of the sets. To that end we show that one can easily compute a set of candidate pairs such that for any isomorphism ϕ of G_1 and G_2 one of the pairs (v_1, v_2) satisfies $\phi(v_1) = v_2$. The runtime of this computation is dominated by the computation of a k-feedback vertex set for G_1, for which the currently fastest FPT-algorithm due to Chen et al. [8] takes time $O(5^k k n^2)$.

Lemma 3. *Let* G_1 *and* G_2 *be two graphs of feedback vertex set number at most* k *and minimum degree at least three. In time* $O(5^k k n^2)$ *one can compute a set* $P \subseteq V(G_1) \times V(G_2)$ *of size at most* $2k + 4k \log k$ *such that (if* G_1 *and* G_2 *are isomorphic) for any isomorphism* ϕ *there is a pair* $(v_1, v_2) \in P$ *such that* $\phi(v_1) = v_2$ *and* v_1 *is contained in a minimum feedback vertex set of* G_1.

Proof. We choose P as $S \times V(C)$ where S is a minimum feedback vertex set of G_1 and C is a shortest cycle of G_2. The time for this computation is dominated by $O(5^k k n^2)$ for computing a k-feedback vertex set of G_1.

If $|V(C)| \leq 6$ then P has size at most $6k$. Otherwise, by Theorem 2, G_2 has at most $2k^2$ vertices. Since the girth of any graph with minimum degree 3 is at most $2\log n$, the cycle C has length at most $2\log(2k^2) = 2 + 4\log k$. Thus P contains at most $2k + 4k\log k$ pairs. Note that $k \geq 2$ for graphs with minimum degree 3 and that $6k \leq 2k + 4k\log k$ for $k \geq 2$.

For any isomorphism ϕ from G_1 to G_2 the image $\phi(S)$ must intersect C since $\phi(S)$ is a feedback vertex set of G_2. Hence P contains a pair (v_1, v_2) with $\phi(v_1) = v_2$ as claimed. ☐

We now design an FPT-algorithm that solves Graph Isomorphism parameterized with the feedback vertex set number. Algorithm 2 performs this task in the following way: Given two graphs of feedback vertex set number at most k, it recursively computes an increasingly large partial isomorphism, given by a set of pairs of vertices $\mathrm{FP} \subseteq V(G_1) \times V(G_2)$. This set indicates that, should an isomorphism exist, there is an isomorphism that maps the first vertex of each pair in FP to the corresponding second vertex. The first components are chosen as to be part of a minimal feedback vertex set in the first graph. At the latest when the set FP has reached size k, removal of the vertices in each graph will result in an OC graph each. Isomorphism can then be decided with the algorithm described in Theorem 3.

Definition 4. We say that an an isomorphism $\phi\colon V(G_1) \to V(G_2)$ *respects a set of pairs of vertices* $\mathrm{FP} \subseteq V(G_1) \times V(G_2)$, if $\phi(v_1) = v_2$ for all $(v_1, v_2) \in \mathrm{FP}$.

We first prove that, given two isomorphic graphs, Algorithm 2 computes an isomorphism.

Lemma 4. *Assume that G_1 and G_2 are two isomorphic graphs of feedback vertex set number at most k. Suppose $k' \in \{0, \dots, k\}$ and $\mathrm{FP} \subseteq V(G_1) \times V(G_2)$ with $|\mathrm{FP}| = k - k'$. Further suppose that there is an isomorphism ϕ from G_1 to G_2 that respects* FP *and that the feedback vertex set number of $R(G_1 - S_1)$ is at most k', where S_1 is the set of first components of the pairs in* FP*. Then the call IsomorphismFVS$(G_1, G_2, k', \mathrm{FP})$ will compute an isomorphism from G_1 to G_2 that respects* FP*.*

Proof. With $\mathrm{FP} = \{(v_{11}, v_{21}), \dots, (v_{1r}, v_{2r})\}$ we define $S_1 = \{v_{11}, \dots, v_{1r}\}$ and $S_2 = \{v_{21}, \dots, v_{2r}\}$. As in the algorithm, let $G_1' = R(G_1 - S_1)$ and let $G_2' = R(G_2 - S_2)$. Since ϕ respects FP, it can be restricted to an isomorphism from $G_1 - S_1$ to $G_2 - S_2$. These graphs are therefore isomorphic and, by Corollary 1, the reduced graphs G_1' and G_2' are isomorphic, under the restriction of ϕ to $V(G_1')$.

We show the lemma by induction on k': If $k' = 0$, the base case, then $G_1' = R(G_1 - S_1)$ is empty since it has feedback vertex set number $k' = 0$. The isomorphic graph G_2' is also empty. The graphs $G_1 - S_1$ and $G_2 - S_2$ are OC graphs by Lemma 2. Therefore the graphs $(G_1, (v_{11}, \dots, v_{1r}))$ and $(G_2, (v_{21}, \dots, v_{2r}))$ are OC+$(k - k')$ graphs and ϕ is an isomorphism of OC+$(k - k')$ graphs. Thus the call to the algorithm described in Theorem 3 will return an isomorphism that respects FP.

If $k' > 0$, we distinguish two cases: Either both G_1' and G_2' are empty, in which case we argue as in the base case, or the algorithm computes P for G_1', G_2', and k' according to Lemma 3. In the set P, since G_1' and G_2' are isomorphic (and non-empty), by Lemma 3, there must be a pair $(v_1, v_2) \in P$ such that $\phi(v_1) = v_2$. Additionally there must be a feedback vertex set of G_1' of size at most k' that contains v_1, implying that the feedback vertex set number of $R(G_1' - v_1)$ is at most $k' - 1$; by Lemma 1 this extends to $R_{S_1 \cup \{v_1\}}(G_1) = R_{\{v_1\}}(G_1') = R(G_1' - v_1)$.

Thus the call IsomorphismFVS$(G_1, G_2, k' - 1, \mathrm{FP} \cup \{(v_1, v_2)\})$ has the property that the isomorphism ϕ respects $\mathrm{FP} \cup \{(v_1, v_2)\}$ and $\mathrm{fvs}(R(G_1 - (S_1 \cup \{v_1\}))) \leq k' - 1$. Hence, by induction, it returns an isomorphism ϕ' that respects $\mathrm{FP} \cup \{(v_1, v_2)\}$. Thus the call IsomorphismFVS$(G_1, G_2, k', \mathrm{FP})$ returns an isomorphism that respects FP, as claimed. ☐

Algorithm 2 IsomorphismFVS

Input: $(G_1, G_2, k, \mathrm{FP})$: An integer k and two graphs G_1, G_2 of feedback vertex set number at most k. A potential partial isomorphism given by pairs of vertices in $\mathrm{FP} \subseteq V(G_1) \times V(G_2)$.

Output: An isomorphism ϕ of G_1 and G_2 that respects FP or report with **false** that no such isomorphism exists.

```
let $(v_{11}, v_{21}), \dots, (v_{1r}, v_{2r})$ denote the pairs in FP
$G'_1 \leftarrow R(G_1 - \{v_{11}, \dots, v_{1r}\})$
$G'_2 \leftarrow R(G_2 - \{v_{21}, \dots, v_{2r}\})$
if if exactly one of $G'_1$ and $G'_2$ is empty then
    return false
end if
if $G'_1$ and $G'_2$ are empty then
    use the algorithm described in Theorem 3 on $(G_1, (v_{11}, \dots, v_{1r}))$ and $(G_2, (v_{21}, \dots, v_{2r}))$
    return an isomorphism $\phi$ with $\phi(v_{1i}) = v_{2i}$ for all $i \in \{1, \dots, r\}$ or false if none exists
end if
if $k = 0$ then
    return false
end if
compute a set $P$ of candidate pairs according to Lemma 3 for $G'_1$, $G'_2$, and $k - r$
for $(v_1, v_2) \in P$ do
    result $\leftarrow$ IsomorphismFVS$(G_1, G_2, k - 1, \mathrm{FP} \cup \{(v_1, v_2)\})$
    if result $\neq$ false then
        return result
    end if
end for
return false
```

The fact that the isomorphism tests for the OC+k graphs are performed in the original input graphs ensures that, even though the reduction R may alter non-isomorphic graphs to be isomorphic (e.g., non-isomorphic trees are reduced to empty graphs), false positives are detected. For this purpose, the partial isomorphism map, encoded by the set FP, has to be maintained by the algorithm, for which Corollary 1 guarantees that it can be lifted into the original graphs and extended to an isomorphism, should it initially arise from an isomorphism.

Now we show that Algorithm 2 is an FPT-algorithm.

Theorem 4. Graph Isomorphism*(*fvs*) is fixed-parameter tractable.*

Proof. Let G_1 and G_2 be two graphs of feedback vertex set number at most k. We show that the call IsomorphismFVS$(G_1, G_2, k, \emptyset)$ correctly determines whether G_1 and G_2 are isomorphic and takes $O((2k + 4k \log k)^k kn^2)$ time.

The algorithm of Theorem 3 will only return valid isomorphisms. IsomorphismFVS will always find an isomorphism if the given graphs are isomorphic, by Lemma 4. It thus suffices to show that the algorithm terminates in the stated time independent of the outcome.

The call IsomorphismFVS$(G_1, G_2, k, \emptyset)$ leads to a recursive computation of depth at most k. The number of recursive calls is limited by the size of P, computed according to Lemma 3, which is bounded by $2k+4k\log k$. The computation at each internal node of the branching tree is dominated by the time necessary for generating P, i.e., by $O(5^k kn^2)$. The calls to the algorithm of Theorem 3 take time $O(n^2)$. This gives the runtime recurrence $T(k) \leq (2k+4k\log k)T(k-1)+O(5^k kn^2)$, which gives a bound of $O((2k+4k\log k)^k kn^2)$.

We conclude that IsomorphismFVS, called as IsomorphismFVS$(G_1, G_2, k, \emptyset)$, is an FPT-algorithm that decides whether G_1 and G_2 are isomorphic. □

Algorithm 2 is also an FPT-algorithm for *colored* graphs with parameter the minimum size of a FVS. For this observe that while Lemma 4 is stated for uncolored graphs it guarantees that for any isomorphism the algorithm finds a corresponding set FP. The algorithm of Theorem 3 will then guarantee that the computed isomorphism respects the colors.

4 Conclusion and open questions

We have shown that Graph Isomorphism is fixed-parameter tractable with respect to the feedback vertex set number. The feedback vertex set number resides above parameters such as the chromatic and the chordal deletion number, with respect to which Graph Isomorphism is not fixed-parameter tractable, unless it may be solved in polynomial time in general. The feedback vertex set number also resides above the parameter treewidth, with respect to which fixed-parameter tractability remains a challenging open problem. In that direction the parameters pathwidth or bandwith are possible further steps to show fixed-parameter tractability of GI with respect to treewidth. Note, that a limited bandwidth simultaneously benefits from a limited treewidth and a limited maximum degree. However even with respect to bandwidth GI might not be fixed-parameter tractable. Showing this, may require a notion of hardness that replaces W[1]-hardness for the not necessarily $\mathcal{NP}$-hard problem Graph Isomorphism (W[1] is a parameterized analogue of $\mathcal{NP}$). The reason for this is that prevalent lower bounds from parameterized complexity, such as W[1]-hardness of GI with respect to some parameter, imply that GI is not in $\mathcal{P}$ unless FPT=W[1].

References

[1] V. Arvind, B. Das, J. Köbler, and S. Toda. Colored hypergraph isomorphism is fixed parameter tractable. *Electronic Colloquium on Computational Complexity (ECCC)*, 16(093), 2009.

[2] L. Babai. Moderately exponential bound for graph isomorphism. In *FCT '81: Proceedings of the 1981 International FCT-Conference on Fundamentals of Computation Theory*, pages 34–50, London, UK, 1981. Springer.

[3] L. Babai, D. Y. Grigoryev, and D. M. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In *STOC '82: Proceedings of the fourteenth annual ACM Symposium on Theory of Computing*, pages 310–324, New York, NY, USA, 1982. ACM.

[4] L. Babai and E. M. Luks. Canonical labeling of graphs. In *STOC '83: Proceedings of the fifteenth annual ACM Symposium on Theory of Computing*, pages 171–183, New York, NY, USA, 1983. ACM.

[5] H. L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11(4):631–643, 1990.

[6] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[7] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

[8] J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences*, 74(7):1188–1198, 2008.

[9] R. G. Downey and M. R. Fellows. *Parameterized Complexity (Monographs in Computer Science)*. Springer, London, UK, November 1998.

[10] S. Evdokimov and I. N. Ponomarenko. Isomorphism of coloured graphs with slowly increasing multiplicity of jordan blocks. *Combinatorica*, 19(3):321–333, 1999.

[11] G. Even, J. Naor, and L. Zosin. An 8-approximation algorithm for the subset feedback vertex set problem. In *FOCS '96: Proceedings of the thirty-seventh annual Symposium on Foundations of Computer Science*, pages 310–319, Washington, DC, USA, 1996. IEEE Computer Society.

[12] H. Fernau. Parameterized algorithms for hitting set: The weighted case. In *CIAC '06: Proceedings of the sixth Italian Conference on Algorithms and Complexity*, pages 332–343, London, UK, 2006. Springer.

[13] I. S. Filotti and J. N. Mayer. A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus. In *STOC '80: Proceedings of the twelfth annual ACM Symposium on Theory of Computing*, pages 236–243, New York, NY, USA, 1980. ACM.

[14] M. Funke and G. Reinelt. A polyhedral approach to the feedback vertex set problem. In *IPCO '96: Proceedings of the fifth Conference on Integer Programming and Combinatorial Optimization*, pages 445–459, London, UK, 1996. Springer.

[15] M. Furst, J. Hopcroft, and E. Luks. Polynomial-time algorithms for permutation groups. In *FOCS '80: Proceedings of the twenty-first annual Symposium on Foundations of Computer Science*, pages 36–41, Washington, DC, USA, 1980. IEEE Computer Society.

[16] M. L. Furst, J. E. Hopcroft, and E. M. Luks. Polynomial-time algorithms for permutation groups. In *FOCS*, pages 36–41. IEEE, 1980.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.

[18] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs. In *STOC '74: Proceedings of the sixth annual ACM Symposium on Theory of Computing*, pages 310–324, New York, NY, USA, 1974. ACM.

[19] K. Kawarabayashi, B. Mohar, and B. A. Reed. A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In *FOCS '08: Proceedings of the fourty-ninth annual Symposium on Foundations of Computer Science*, pages 771–780, Washington, DC, USA, 2008. IEEE Computer Society.

[20] D. E. Knuth, J. H. Morris Jr., and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.

[21] E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.

[22] D. Marx. Chordal deletion is fixed-parameter tractable. In *WG '06: 32nd International Workshop on Graph-Theoretic Concepts in Computer Science, Revised Papers*, pages 37–48, London, UK, 2006. Springer.

[23] G. L. Miller. Isomorphism testing for graphs of bounded genus. In *STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 225–235, New York, NY, USA, 1980. ACM.

[24] I. N. Ponomarenko. The isomorphism problem for classes of graphs closed under contraction. *Journal of Mathematical Sciences*, 55(2):1621–1643, 1991.

[25] V. Raman, S. Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Transactions on Algorithms*, 2(3):403–415, 2006.

[26] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323, 1988.

[27] R. E. Tarjan. A V^2 algorithm for determining isomorphism of planar graphs. *Information Processing Letters*, 1(1):32–34, 1971.

[28] S. Thomassé. A quadratic kernel for feedback vertex set. In *SODA '09: Proceedings of the twentieth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 115–119. SIAM, 2009.

[29] S. Toda. Computing automorphism groups of chordal graphs whose simplicial components are of small size. *IEICE Transactions*, 89-D(8):2388–2401, 2006.

[30] R. Uehara, S. Toda, and T. Nagoya. Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. *Discrete Applied Mathematics*, 145(3):479–482, 2005.

[31] K. Yamazaki, H. L. Bodlaender, B. de Fluiter, and D. M. Thilikos. Isomorphism for graphs of bounded distance width. *Algorithmica*, 24(2):105–127, 1999.