On Tractable Parameterizations of Graph Isomorphism

Adam Bouland[1] and Anuj Dawar[2] and Eryk Kopczyński[3]

[1] Massachusetts Institute of Technology, Cambridge, MA, USA
[2] University of Cambridge, UK
[3] University of Warsaw, Poland

Abstract. The fixed-parameter tractability of graph isomorphism is an open problem with respect to a number of natural parameters, such as tree-width, genus and maximum degree. We show that graph isomorphism is fixed-parameter tractable when parameterized by the *tree-depth* of the graph. We also extend this result to a parameter generalizing both tree-depth and max-leaf-number by deploying new variants of cops-and-robbers games.

1 Introduction

The fixed-parameter complexity of the graph isomorphism problem (GI) remains open with respect to a number of interesting graph parameters. Several parameterizations of graph isomorphism are known to yield tractable algorithms. For instance, graph isomorphism is known to be fixed-parameter tractable in the following parameters: size of the smallest feedback vertex set [17], tree-distance width [28], largest multiplicity of an eigenvalue of the adjacency matrix [8], size of the largest color class (in the case of colored graph isomorphism) [2][10][1], and maximum size of a simplicial component (in the case of chordal graph isomorphism) [27].

On the other hand, many natural parameterizations of the problem are known to produce algorithms which run in time $O(n^{f(k)})$, which places them in XP, but for which fixed-parameter tractability remains open. For instance, graph isomorphism is in XP when parameterized by the size of the smallest excluded minor [25][14] or topological minor [13] of a graph . This generalizes a long line of previous results that GI is in XP with respect to a number of other parameters, including genus [22], maximum degree [20], and tree-width [4]. It should be pointed out that in none of these cases do we know of any hardness result that indicates the problem is not fixed-parameter tractable.

One particular open question is whether or not GI is fixed-parameter tractable when parameterized by tree-width or path-width. In the present paper, we show that the problem is fixed-parameter tractable when parameterized by the *tree-depth* of a graph. The tree-depth of a graph measures how close a graph is to a star, in much the same way that tree-width measures how close a graph is to a tree. This parameter is natural in the context of sparse matrix factorization

[15][21] and descriptive complexity [7]. Our proof yields a natural generalization to a parameter we introduce and call *generalized tree-depth*, which generalizes the parameter *max-leaf-number* as well.

The key idea in our proof is to use a characterization of tree-depth in terms of cops and robbers games in order to show that in any graph G of tree-depth at most d, the number of vertices that can serve as "roots" of a minimum height tree-depth decomposition is bounded by a function of d. This allows us to create an automorphism-invariant tree-depth decomposition algorithm based on Lindell's algorithm for logspace tree canonization [18].

2 Preliminaries

A language Q over an alphabet Σ is said to be *fixed-parameter tractable* with respect to parameterization $\kappa : \Sigma^* \to \mathbb{N}$, if it can be decided on input x in time $O(f(k)n^c)$ where $n = |x|$, c is a fixed constant, $k = \kappa(x)$ is the value of the parameter and f is an arbitrary function.

The *max-leaf-number* of a graph G is the maximum number of leaves in a spanning tree of G.

The *tree-depth* of a graph is defined recursively as follows:

Definition 1. *Let G be a graph with connected components G_1, ..., G_p. Then the tree-depth of G, denoted* $\mathrm{td}(G)$*, is given by*

$$\mathrm{td}(G) = \begin{cases} 1 & \text{if } |V(G)| = 1 \\ 1 + \min\limits_{v \in V(G)} \mathrm{td}(G - v) & \text{if } p = 1 \text{ and } |V(G)| > 1 \\ \max\limits_{i=1\ldots p} \mathrm{td}(G_i) & \text{otherwise} \end{cases}$$

For example, the tree-depth of a star is 2, and the tree-depth of the complete graph K_n is n. In some sense tree-depth measures how close a graph is to a star. Tree-depth occurs naturally in descriptive complexity, in which it was recently shown that monadic second order logic and first order logic coincide on a class of graphs C iff C has bounded tree-depth [7].

An alternative definition of tree-depth is also helpful for our results. The *height* of a rooted tree T is the length of the longest path from the root to a leaf. The *closure* of a rooted tree T, denoted $\mathrm{clos}(T)$, is the graph obtained by adding edges from each vertex v to all vertices w which lie on a path from the root to v. Then the tree-depth of a connected graph G is the minimum height of a tree such that G is a subgraph of $\mathrm{clos}(T)$ [24].

Consider a connected graph G and a tree T over $V(G)$ such that G is a subgraph of $\mathrm{clos}(T)$. We will call such a tree T a *tree-depth decomposition* of G if it obeys the following property: for every rooted subtree of T, the subgraph of G induced by the subtree is connected. Here by a rooted subtree, we mean a subtree induced by a single node v and all of its children in T. It can be easily shown that a graph has tree-depth $\leq d$ iff it has a tree-depth decomposition of depth d. The *root* of the decomposition is the root of the tree T. Note that

if G is disconnected, we define the tree-depth decomposition as a rooted forest consisting of the tree-depth decompositions of its connected components.

Tree-depth is well-behaved with respect to the operation of taking minors. Also, we can test if a graph has tree-depth d efficiently:

Claim 2. *If H is a minor of G, then $\mathrm{td}(H) \leq \mathrm{td}(G)$ [24].*

Claim 3. *Given a graph G, we can find the tree-depth of G in time $O(f(d)n^2)$ for some computable function f, where $d = \mathrm{td}(G)$ [23].*

We note that the tree-depth of a graph gives a lower bound on the vertex cover number of G. To see this, note that any graph of vertex cover number k has a depth $k+1$ decomposition tree T taken as follows: Order the vertices of a minimal vertex cover arbitrarily and place them in a path. At the bottom of the path, attach all remaining nodes of the graph as leaves. This obeys $E(G) \subseteq \mathrm{clos}(T)$ by the definition of vertex cover, so for any graph G, $\mathrm{td}(G) \leq \mathrm{vcn}(G) + 1$.

Likewise, it can be easily shown that the path-width of a graph is a lower-bound on its tree-depth [3]. So if GI were FPT when parameterized by path-width, then it would trivially also be FPT parameterized by tree-depth. However GI is not known to be fixed-parameter tractable when parameterized by path-width.

3 Games

Suppose that G is a connected graph of tree-depth d. We will show that the number of vertices in G which can serve as a root of a minimal tree-depth decomposition is bounded as a function of d. In order to prove this result, we will descibe yet another equivalent definition of tree-depth in terms of cops and robbers games. Such games are frequently used in the context of logic and graph isomorphism, e.g. [5]. The bound we obtain on the number of roots will play a crucial role in our isomorphism algorithm.

3.1 A Characterization of Tree-Depth in Terms of Cops-and-Robbers

We define a cops-and-robbers game in which the cops do not move once they land on the graph. Thus, the number of moves in the game is limited by the number of cops. We make this precise below.

Consider the following game played on a connected graph $G = (V, E)$. The game is played by two players, called Cop and Robber. The Cop player controls d cops, and the Robber player controls one robber.

First, Robber places the robber on any vertex in G, and announces his position. Cop announces a position where he will place his next cop. In response, the robber can move along a path in the graph to another position, including the one announced by Cop, but he cannot move through positions previously occupied by cops.

The Cop player wins if, at the end of the move, the robber is on the vertex just occupied by a cop, and the Robber player wins if all d cops are on the graph and the robber is still not caught. This game is known to capture tree-depth [11][12] in the following way:

Claim 4. *For a connected graph G, $\mathrm{td}(G)$ is equal to the least d for which the Cop player has a winning strategy.*

Proof. If $\mathrm{td}(G) = d$, consider the following strategy for the Cop player: place the first cop on the root r of the decomposition. Place the next cop on the root of the depth $d-1$ decomposition of the connected component of $G - r$ containing the robber. Repeat. Clearly this is a winning strategy for the Cop player with d cops. On the other hand, given a winning strategy γ with d cops, construct a decomposition by taking the root of each (sub)decomposition to be the position played by γ if the robber is in that connected component. Since γ uses d cops, the depth of the resulting tree T is at most d, and we will have $E \subseteq \mathrm{clos}(T)$ because γ is a winning strategy. □

Hence a vertex v is a *root* of a tree-depth decomposition of minimal depth iff the Cop player has a winning strategy using $\mathrm{td}(G)$ cops which places the first cop at v. Let $\mathrm{root}(G)$ be the set of all roots. In the rest of this section we will provide a self-contained proof that $|root(G)|$ is bounded by a function of $d = \mathrm{td}(G)$. As pointed out by an anonymous referee, this fact also follows easily from a recent paper by Dvořák, Giannopoulou and Thilikos [6]. These authors showed that the class $C_d = \{G : \mathrm{td}(G) \leq d\}$ is characterized by a finite set of forbidden subgraphs, each with at most $2^{2^{d-1}}$ vertices. Now consider a graph G of tree-depth d. Since $G \notin C_{d-1}$, there exists a subgraph H of G containing at most $2^{2^{d-2}}$ vertices with $\mathrm{td}(H) = d$. If γ is a winning strategy for Cop on G using at most d cops, then γ must make its first move in H. Indeed, if γ makes its first move outside of H, then the robber player can move into H, and subsequently play an optimal Robber strategy for H, forcing γ to use $d+1$ cops to win. Therefore $\mathrm{root}(G) \subseteq H$, so $|\mathrm{root}(G)| \leq 2^{2^{d-2}}$. We conjecture that $|\mathrm{root}(G)| = 2^{O(d)}$, but for our purposes it is enough to show that it is bounded.

3.2 Components and Isomorphisms

Consider the state of the game after k rounds of play. Let B be the set of k vertices occupied by cops so far.

We say that $C \subseteq V$ is a *component* of $V - B$ if there are no edges between C and $V - C - B$, i.e., the Cops have blocked all exit routes from C. We say that two components C_1 and C_2 are *isomorphic* iff there is a bijection $\phi : C_1 \cup B \to C_2 \cup B$ such that $\phi(b) = b$ for $b \in B$, and $E(v_1, v_2)$ iff $E(\phi(v_1), \phi(v_2))$.

3.3 Counting Components

We will show that for a connected graph G, $\mathrm{td}(G)$ and $\mathrm{root}(G)$ are unaffected by removing "extra" copies of isomorphic components which arise in the course of

the game. This will be the key fact which allows us to bound the size of root(G) as a function of the tree-depth.

Lemma 5. *Let G be a connected graph with* $\mathrm{td}(G) = d$. *Let $B \subseteq V$ be a set of k vertices, and let $C_1, C_2, \ldots, C_u$ be isomorphic components of $V - B$, where $u \geq d+1$. Let G' be the graph obtained from G by removing all the components C_i for $i > d+1$. Then* $\mathrm{td}(G) = \mathrm{td}(G')$ *and* $\mathrm{root}(G) = \mathrm{root}(G')$.

Proof. Without loss of generality we can assume that for each $b \in B$ there is an edge between b and C_i.

Let ρ be a Robber strategy which forces Cop to use d cops. We will construct a Robber strategy ρ' based on ρ which forces the Cop player to use at least d cops on G'. This will show $\mathrm{td}(G') \geq d$. Since G' is a subgraph of G, $\mathrm{td}(G') \leq d$, so this will show the tree-depth is unaffected by removing the extra copies of isomorphic components.

Let γ'' be a Cop strategy on G'. We will play γ'' against ρ', and construct ρ' to force γ'' to use d cops.

We start by having ρ' place the robber on an arbitrary vertex of the graph (it does not matter since the graph is connected). Then we construct ρ' in two stages. The basic idea is to mirror the strategy ρ as closely as possible. We will only have to change the strategy if γ'' plays in a C_i for $i \geq d+1$ (because we deleted these vertices), or once B has been filled with cops.

The first stage begins, and ends iff cops have been placed on all vertices of B, and the robber has moved to a C_i. This ensures that throughout this stage, the robber can move between all copies of C_i in G' which do not contain cops. We now have ρ' play the same move that ρ would play in G, unless ρ plays in a C_i for $i \geq d+1$. If this occurs, we mimic the response of ρ via the isomorphism in one of the copies of C_i in G' which currently does not contain any cops. Since any Cop player on G can place cops in at most d copies of C_i, and we have kept d copies of the C_i in G', we will never run out of copies to mirror this strategy.

If γ'' never exits the first stage, it must use at least d cops to win. Indeed if the robber is connected to the C_i's which contain no cops, the robber can always move between these C_i's via B, so γ'' loses. If the robber is confined outside the C_i's, γ'' must use at least d cops because ρ' is identical to ρ once it is confined outside the C_i's.

If γ'' does place cops on all of B, with the robber confined to a C_i, we proceed to the second stage. In this stage we simply directly copy the behavior of ρ on an isomorphic copy of C_i as before. We can easily see that ρ' forces the Cop player to use d cops, and hence $\mathrm{td}(G) = \mathrm{td}(G')$.

Now to show $\mathrm{root}(G) = \mathrm{root}(G')$, consider any winning Cop strategy γ' induced by a winning strategy γ on G. If γ makes its first move on a C_i, then we could remove this C_i in constructing G' to create a winning strategy for the Cop player on G' using $d-1$ cops, which is a contradiction. Hence $\mathrm{root}(G)$ must be disjoint with C_i for all i. Therefore γ must place its first cop outside all C_i, and so does γ'. Therefore $root(G) \subseteq root(G')$. We can likewise reverse this entire argument by considering adding copies of isomorphic components to G' to obtain

a larger graph G, assuming G' has at least d copies of the component already. By constructing the Cop and Robber strategies for G based on the strategies for G', we can see that $\text{root}(G') \subseteq \text{root}(G)$. Thus $\text{root}(G) = \text{root}(G')$. □

3.4 Measuring components

Let G be an arbitrary graph. As long as we can find a set $B \subseteq V(G)$ such that the graph $G - B$ contains more than $d+1$ isomorphic components, we remove the extra components by Lemma 5. Ultimately we obtain a *minimal* graph G' where each component appears at most $d+1$ times. From Lemma 5 we know that $\text{root}(G') = \text{root}(G)$. Thus, we have only to show that $|\text{root}(G)|$ is bounded for minimal graphs.

Let γ be winning a strategy for the Cop player which uses at most d cops, and let ρ be any robber strategy. Then the following holds.

Lemma 6. *Let G be a connected graph which is minimal as described in Lemma 5. Let B be the set of vertices blocked by cops after i rounds of play between any such γ and ρ. Then there exists a function f such that the component of $G - B$ containting the robber consists of at most $f(d, i)$ vertices.*

Proof. The proof follows by reverse induction on i. For $i = d$ we know that Robber has been caught, so $f(d, d) = 0$.

For $i < d$, let v be the vertex where γ puts its next cop. Let $B' = B \cup \{v\}$. From the inductive assumption we know that each component of $V - B'$ has size at most $s = f(d, i+1)$. Up to isomorphism there are at most $S = 2^{\binom{s}{2}(i+1)}$ possible components of size s. Since the graph G is minimal, each of them appears at most $d+1$ times. Thus, $f(d, i) \leq 1 + (d+1)S$. □

Thus, a minimal graph of tree-depth d has at most $f(d, 0)$ vertices, and we have proven the following lemma:

Lemma 7. *If a connected graph G has tree-depth d, then the number of roots of tree-depth decompositions of G of minimal depth is at most $f(d)$ for some function of d.*

4 Isomorphism Algorithm

We will now create an algorithm which shows that graph isomorphism parameterized by tree-depth is in FPT. The basic idea is to extend the logspace algorithm for tree isomorphism developed by Lindell [18] to test for isomorphism over tree-depth decompositions.

Lindell's algorithm works by establishing an ordering $<$ on the set of connected trees [18]. In his algorithm, two trees S and T obey $S < T$ if

1. $|S| < |T|$, where $|S|$ denotes the number of nodes in S.
2. $|S| = |T|$ and $\#s < \#t$, where $\#s$ is the number of children of the root of S

3. $|S| = |T|$, $\#s = \#t = k$ and $(S_1, S_2, ..., S_k) < (T_1, T_2, ..., T_k)$ lexicographically, where we inductively assume that $S_1 \leq S_2 \leq ... \leq S_k$ and $T_1 \leq T_2 \leq ... \leq T_k$ are the ordered subtrees of S and T obtained by removing the roots of S and T

Clearly $S \cong T$ iff neither $S < T$ nor $T < S$ [18].

We will extend this ordering on trees to an ordering of the tree-depth decompositions. To test for isomorphism, we will find a minimal, canonical decomposition of each graph and compare the decompositions.

Recall that a tree-depth decomposition of a connected graph G consists of a rooted tree T over $V(G)$ such that $E(G) \subseteq E(\text{clos}(T))$. A tree-depth decomposition also has the property that any induced subgraph of G obtained by the vertices of a rooted subtree of T is connected.

We say that two decompositions T_1 of G_1 and T_2 of G_2 are equivalent, denoted $T_1 \simeq T_2$, if there is an isomorphism ϕ between T_1 and T_2 which preserves the edges of the underlying graphs as well, i.e. both $(u, v) \in E(T_1) \Leftrightarrow (\phi(u), \phi(v)) \in E(T_2)$ and $(u, v) \in E(G_1) \Leftrightarrow (\phi(u), \phi(v)) \in E(G_2)$. In particular, $T_1 \simeq T_2$ implies $G_1 \cong G_2$.

Suppose that we are given a connected graph G with $\text{td}(G) = d$. Given a tree-depth decomposition T of G, define a sub tree-depth decomposition of a graph G' induced by a subtree of T' of T to consist of the following: the tree T' over $V(G')$ with root t', as well as the path P from the parent of t' to the root of T. If $td(G') = d'$, then P consists of vertices $r = r_1, r_2, ...r_{d-d'}$, where r is the root of T and $r_{d-d'}$ is the parent of t' in T. See Figure 1 for clarification.

We inductively define an ordering of sub tree-depth decompositions of G as follows. Let S and T be two depth d' subdecompositions of G_S and G_T, respectively, with roots s and t, respectively, and which share the same path P defined above. Note S and T must share the same path P to be comparable. Also note that when considering the entire graph, P is empty so this defines an ordering on all tree-depth decompositions.

We say that the subdecomposition S of G_S is less than the subdecomposition T of G_T, denoted $S < T$, if one of the following conditions is satisfied:

1. $|G_S| < |G_T|$
2. $|G_S| = |G_T|$ and $\#s < \#t$, where $\#x$ is the number of connected components in $G_X - x$.
3. $|G_S| = |G_T|$, $\#s = \#t$, and $(E(r_1, s), E(r_2, s), ..., E(r_{d-d'}, s)) < (E(r_1, t), E(r_2, t), ..., E(r_{d-d'}, t))$ lexicographically, where $E(x, y) = 1$ if there is an edge from x to y and 0 otherwise. If $d' = d$ this condition is trivially satisfied.
4. $|G_S| = |G_T|$, $\#s = \#t$, $E(r_i, s) = E(r_i, t)$ $\forall i = 1...(d - d')$ and

$$(S_1, S_2, ..., S_k) < (T_1, T_2, ..., T_k)$$

lexicographically, where we inductively assume $S_1 \leq S_2 \leq ... \leq S_k$ and $T_1 \leq T_2 \leq ... \leq T_k$ are the connected components of $G_S - s$ and $G_T - t$, ordered by their subdecompositions induced by S and T. (Here $S \leq T$ means $S < T$ or $S \simeq T$).

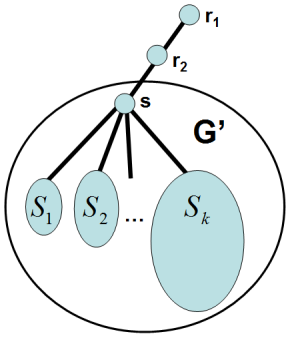

Fig. 1. A sub-decomposition of G' with root s and components $S_1 ... S_k$ of $G' - s$.

This ordering has several nice properties. The following can be shown by simple induction on the tree-depth:

Claim 8. *Suppose G and H are connected graphs, both of tree-depth d and the same size. Let S be a minimal tree-depth decomposition of G and T a minimal tree-depth decomposition of H according to the above ordering. Then if neither $S < T$ nor $T < S$, then $S \simeq T$ and $G \cong H$.*

By condition (4) of the ordering, we know that to find the minimal decomposition of G rooted at s, we simply need to find the minimal decompositions of each of the connected components of $G - s$. This forms the basis of a recursive algorithm to compute the minimum depth-d decomposition of a graph G, Algorithm 1. With this in hand, we can show that Algorithm 2 correctly tests for isomorphism.

Algorithm 1: Recursive construction of a minimal tree-decomposition

Input: A connected graph G' of tree-depth d' along with a specified path $P = r_1 ... r_l$.

Output: A sub tree-depth decomposition S of G' of depth d' which is minimal with respect to $<$ for P.

if $\mathrm{td}(G) = 1$ **then**
  Output the trivial decomposition of the graph.
else
  Find $R = \{v \in V(G') : \mathrm{td}(G' - v) = d - 1\}$.
  Remove those elements $r \in R$ which do not have minimal values of $(E(r_1, r), E(r_2, r), ..., E(r_l, r))$ or $\#r$.
  foreach $r \in R$ **do**
    Compute minimal decompositions of $S_1 ... S_k$, the connected components of $G' - r$, using this algorithm and appending r path P.
    Order $S_1 ... S_k$ by $<$ using $k \log k$ comparisons.
  end
  Find which $r \in R$ produces the decompositions $(S_1, ... S_k)$ which are minimal in lexicographic order, and output the decomposition obtained by making this the root of the decomposition.
end

Algorithm 2: An isomorphism algorithm parameterized by tree-depth

Input: Two graphs G and H.
Output: Whether or not $G \cong H$.
Check that $\mathrm{td}(G) = \mathrm{td}(H) = d$, if not output that G and H are not isomorphic.
Compute S, a minimal decomposition of G , and T, a minimal decomposition of H using Algorithm 1 with an empty P.
If neither $S < T$ nor $T < S$, output $G \cong H$.
Else output that G and H are not isomorphic.

Claim 9. *If $G \cong H$ are connected graphs, then the decompositions produced by Algorithm 1 on G and H are isomorphic.*

Proof. Follows because all steps in Algorithm 1 are isomorphism invariant. □

Corollary 10. *Algorithm 2 correctly tests for isomorphism over connected graphs.*

Theorem 11. *Graph isomorphism is fixed-parameter tractable in tree-depth.*

Proof. We will upper bound $T(n, d)$, the runtime of Algorithm 1 on a connected graph G with n vertices and tree-depth d.

By Claim 3 we can check if $\mathrm{td}(G) = d - 1$ in time $f(d-1)n^2$, so finding $R = \{v \in V(G') : \mathrm{td}(G' - v) = d - 1\}$ can be done in time $f(d)n^3$.

Next we reduce the size of R to only those vertices with minimal $\#r$. For each r, computing $\#r$ can be done in time $\Sigma_{i=1\ldots\#r} O(|S_i|^2) \leq O(n^2)$. Hence this step takes time $O(g(d)n^2)$, since by Lemma 7 $|R| \leq g(d)$.

Now the algorithm recurses. By Lemma 7 it will recurse on at most $g(d)$ different roots. For each of these roots r, if $k = \#r$ then it will compute a decomposition of each of the connected components $S_1 \ldots S_k$ of $G - r$, which will take time $\leq \Sigma_{i=1}^{k} T(|S_i|, d-1)$. To order these decompositions by $<$, it will then make $k \log k$ comparisons of the decompositions using $<$, each of which takes time $O(n^2)$, and subsequently make $g(d)k$ comparisons in time $g(d)kn^2$ between these sorted decompositions to find which of the $g(d)$ roots is minimal.

This yields a recursion relation for the run time given by

$$T(n,d) \leq f(d)n^3 + g(d)n^2 + g(d)\left\{\left(\sum_{i=1}^{k} T(|S_i|, d-1)\right) + O(k \log k n^2)\right\} \quad (1)$$

One can easily check that a run time of $T(n, d) = h(d)n^3 \log(n)$ suffices. Plugging this ansatz into the recursion relation, and simplifying using the convexity of $n^3 \log(n)$ and the fact that $k \leq n$, one can see that

$$T(n,d) \leq (f(d) + g(d))n^3 \log(n) + g(d)\left\{h(d-1)n^3 \log(n) + O(n^3 \log(n))\right\} \quad (2)$$

Taking $h(d) = f(d) + g(d)\left(h(d-1) + O(1)\right)$, we have $T(n, d) \leq h(d)n^3 \log(n)$. By setting $h(0) = 1$, this provides an inductive definition of h as a function.

Therefore Algorithm 1 runs in time $O(h(d)n^3 \log(n))$. Since checking if two decompositions are equivalent takes $O(n^2)$ time (assuming the orderings of the sub-decompositions of each level are recorded), Algorithm 2 correctly decides isomorphism over connected graphs in time $O(h(d)n^3 \log(n))$. This algorithm extends easily to disconnected graphs by the convexity of n^3. □

5 Generalized Tree-Depth

We will now define a new parameter which generalizes both tree-depth and max-leaf-number. Recall that the max-leaf-number of a graph G, denoted $\mathrm{mln}(G)$, is the maximum number of leaves in a spanning tree of G. A crucial fact is that if $\mathrm{mln}(G) = k$, then the number of vertices of degree $\neq 2$ in G is bounded by a function of k. This can be easily used to show that graph isomorphism is fixed-parameter tractable in max-leaf-number, by simply trying all bijections between vertices of degree $\neq 2$ and noting that all other vertices lie on simple paths.

Claim 12 (From Kleitman and West [16] via [9]). *If G is a graph with $\mathrm{mln}(G) \leq k$, then G is a subdivision of a graph H with at most $4k-2$ vertices.*

Low max-leaf-number means that the graph becomes a collection of paths after removing a small number of vertices. Low tree-depth, on the other hand, means that the graph quickly degenerates to an empty set after alternately removing vertices and considering disjoint components separately. Following the example of max-leaf number, we can generalize tree-depth by allowing a broader class of graphs as leftovers at the end of k-cops and robbers game. Previously, we said that the Cop player wins the game if a cop lands on the robber. Consider a modified version of the game, in which Cop wins if the robber is confined to either a simple path with cops at both endpoints or a simple cycle. The endpoints of this path (which must be occupied by cops) may be connected to other vertices of the graph, but the other points of the path may not have any edges to the rest of the graph. To win by confining the robber to a cycle, the cycle must be disconnected from the rest of the graph.

Definition 13. *A graph G has generalized tree-depth d, denoted $\mathrm{gtd}(G) = d$, iff d is the least k such the Cop player has a winning strategy in the modified k-cops and robber game described above.*

It is clear that the generalized tree-depth of a graph is a lower bound on its tree-depth, since a single vertex is a simple path of length zero. Furthermore, this parameter bounds from below the max-leaf-number, because the Cop player has a winning strategy using $4\mathrm{mln}(G)-2$ cops by placing places cops on all vertices of degree $\neq 2$. Therefore for any graph G, $\mathrm{gtd}(G) \leq 4\mathrm{mln}(G)$ and $\mathrm{gtd}(G) \leq \mathrm{td}(G)$, so this parameter generalizes both tree-depth and max-leaf-number.

We can now extend our arguments from the previous sections to show that graph isomorphism is fixed-parameter tractable in the generalized tree-depth. First, note that having generalized tree-depth d is equivalent to having a tree-depth decomposition of depth d as before, except that the leaves of the tree

can now consist of simple paths. The simple cycles, being disconnected from the graph, are omitted from the decomposition and are handled separately later. The endpoints of the path in each leaf are specified in the decomposition. By the same arguments as in [24], $C_d = \{G : \mathrm{gtd}(G) \leq d\}$ is closed under taking minors for $d \geq 2$, while C_0 and C_1 have trivial poly-time membership tests, yielding a (non-constructive) FPT algorithm to compute generalized tree-depth in time $O(f(d)n^3)$ by the Robertson-Seymour Theorem [26][19].

Next, note that our arguments bounding the number of roots of a decomposition also carry through. When counting the number of vertices in each component, we count only the endpoints of the paths in the leaves, since only these vertices can connect to the rest of the graph. These specified endpoints are considered the roots of the leaf's decomposition. This ensures that the base case of Lemma 6 is still bounded above by two. We again obtain that the number of roots of a graph of generalized tree-depth d is bounded by a function $g(d)$. The fact that we handle simple cycles separately is crucial to keeping this bound.

We can likewise extend Lindell's tree isomorphism algorithm to an FPT algorithm for generalized tree-depth exactly as before. To do so, we simply modify our ordering on decompositions to take in to account the number of nodes in the path of each leaf, and handle the simple cycles separately. This yields an algorithm to test for isomorphism in $O(h(d)n^4)$ time, where d is the generalized tree-depth and $h(d)$ is a function which is not necessarily computable, as we have used the Robertson-Seymour theorem. We have thus shown:

Theorem 14. GI *is fixed-parameter tractable in generalized tree-depth.*

6 Conclusion

We have shown that graph isomorphism is fixed-parameter tractable when parameterized by generalized tree-depth. An open question is whether or not GI is FPT in the path-width of the graph. Unlike in the case of tree-depth, the number of valid path-width decompositions of a graph is exponential in the number of vertices, so our approach does not immediately generalize.

Acknowledgments. Research by Adam Bouland was partially supported by the NSF Graduate Research Fellowship under grant no. 1122374 and by a Marshall Scholarship. Research by Anuj Dawar was partially supported by EPSRC grant EP/H026835. Research by Eryk Kopczyński was partially supported by ESF Research Networking Programme GAMES and by the Polish National Science Centre (grant N N206 567140).

References

1. Arvind, V., Das, B., Johannes, K., Toda, S.: Colored Hypergraph Isomorphism is Fixed Parameter Tractable. ECCC 93 (2009)

2. Babai, L.: Monte-Carlo algorithms in graph isomorphism testing. Tech. Rep. DMS 79-10, Université de Montréal pp. 1–33 (1979)
3. Bodlaender, H., Hafsteinsson, H., Gilbert, J.R., Kloks, T.: Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. J. Algorithms 18, 238–255 (1995)
4. Bodlaender, H.L.: Polynomial Algorithms for Graph lsomorphism and Chromatic Index on Partial k-Trees. Journal of Algorithms 11(4), 631–643 (1990)
5. Cai, J.Y., Fürer, M., Immerman, N.: An Optimal Lower Bound on the Number of Variables for Graph Identification. Combinatorica 12(4), 389–410 (1992)
6. Dvořák, Z., Giannopoulou, A., Thilikos, D.M.: Forbidden graphs for tree-depth. European Journal of Combinatorics 33(5), 969–979 (2012)
7. Elberfeld, M., Grohe, M.: Where First-Order and Monadic Second-Order Logic Coincide. Arxiv preprint arXiv:1204.6291 pp. 1–15 (2012)
8. Evdokimov, S., Ponomarenko, I.: Isomorphism of Coloured Graphs with Slowly Increasing Multiplicity of Jordan Blocks. Combinatorica 19(3), 321–333 (1999)
9. Fellows, M., Lokshtanov, D., Misra, N., Mnich, M., Rosamond, F., Saurabh, S.: The Complexity Ecology of Parameters: An Illustration Using Bounded Max Leaf Number. Theory of Computing Systems 45(4), 822–848 (2009)
10. Furst, M., Hopcroft, J., Luks: Polynomial-time algorithms for permutation groups. In: Proc. FOCS 1980 pp. 36–41 (1980)
11. Ganian, R., Hliněný, P., Kneis, J., Langer, A., Obdrzalek, J., Rossmanith, P.: On digraph width measures in parameterized algorithmics. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917 pp. 185-197. Springer, Heidelberg (2009)
12. Giannopoulou, A., Hunter, P., Thilikos, D.: LIFO-search: A min-max theorem and a searching game for cycle-rank and tree-depth. Submitted to J. Discrete Math. (2011)
13. Grohe, M., Marx, D.: Structure Theorem and Isomorphism Test for Graphs with Excluded Topological Subgraphs. In: Proc. STOC 2012 pp. 173–192 (2012)
14. Grohe, M.: Fixed-point definability and polynomial time on graphs with excluded minors. In: Proc. LICS 2010 pp. 179–188 (2010)
15. Heath, M., Ng, E., Peyton, B.: Parallel algorithms for sparse linear systems. SIAM review 33(3), 420–460 (1991)
16. Kleitman, D., West, D.: Spanning Trees with Many Leaves. SIAM J. Discrete Math. 4, 99–106 (1991)
17. Kratsch, S., Schweitzer, P.: Isomorphism for graphs of bounded feedback vertex set number. In: Kaplan, H. (ed.) SWAT 2010. LNCS, vol. 6139, pp. 81-92. Springer, Heidelberg (2010)
18. Lindell, S.: A logspace algorithm for tree canonization. In: Proc. STOC 1992 pp. 400–404 (1992)
19. Lovász, L.: Graph minor theory. Bulletin of the AMS 43(1), 75–86 (2006)
20. Luks, E.: Isomorphism of graphs of bounded valence can be tested in polynomial time. Journal of Computer and System Sciences (1982)
21. Manne, F.: An Algorithm for Computing an Elimination Tree of Minimum Height for a Tree. Tech. Rep. CS-91-59, University of Bergen, Norway (1992)
22. Miller, G.: Isomorphism testing for graphs of bounded genus. In: Proc. STOC 1980 pp. 225–235 (1980)
23. Nešetřil, J., Ossona de Mendez, P.: Sparsity: Graphs, Structures and Algorithms, Algorithms and Combinatorics, vol. 28. Springer (2012)
24. Nešetřil, J., Ossona de Mendez, P.: Tree-depth, subgraph coloring and homomorphism bounds. European Journal of Combinatorics 27(6), 1022–1041 (2006)

25. Ponomarenko, I.: The isomorphism problem for classes of graphs that are invariant with respect to contraction (Russian). Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI) 174, 147–177 (1988)
26. Robertson, N., Seymour, P.: Graph minors XX. Wagners conjecture. Journal of Combinatorial Theory, Series B 92, 325–357 (2004)
27. Toda, S.: Computing automorphism groups of chordal graphs whose simplicial components are of small size. IEICE Transactions on Information and Systems E89-D(8), 2388–2401 (2006)
28. Yamazaki, K., Bodlaender, H.L., de Fluiter, B., Thilikos, D.M.: Isomorphism for Graphs of Bounded Distance Width. Algorithmica 24(2), 105–127 (1999)