Symposium on Theoretical Aspects of Computer Science 2010 (Nancy, France), pp. 35-46
www.stacs-conf.org

LARGE-GIRTH ROOTS OF GRAPHS

ANNA ADAMASZEK [1] AND MICHAŁ ADAMASZEK [2]

[1] Department of Computer Science and DIMAP,
University of Warwick, Coventry, CV4 7AL, UK
E-mail address: annan@mimuw.edu.pl

[2] Warwick Mathematics Institute and DIMAP,
University of Warwick, Coventry, CV4 7AL, UK
E-mail address: aszek@mimuw.edu.pl

Abstract. We study the problem of recognizing graph powers and computing roots of graphs. We provide a polynomial time recognition algorithm for r-th powers of graphs of girth at least $2r+3$, thus improving a bound conjectured by Farzad et al. (STACS 2009). Our algorithm also finds all r-th roots of a given graph that have girth at least $2r+3$ and no degree one vertices, which is a step towards a recent conjecture of Levenshtein that such root should be unique. On the negative side, we prove that recognition becomes an NP-complete problem when the bound on girth is about twice smaller. Similar results have so far only been attempted for $r = 2, 3$.

1. Introduction

All graphs in this paper are simple, undirected and connected. If H is a graph, its *r-th power* $G = H^r$ is the graph on the same vertex set such that two distinct vertices are adjacent in G if their distance in H is at most r. We also call H the *r-th root* of G.

There are some problems naturally related to graph powers and graph roots. Suppose $\mathcal{P}$ is a class of graphs (possibly consisting of all graphs), r is an integer and G is an arbitrary graph. The questions we ask are:

- *The recognition problem*: Is G an r-th power of some graph from $\mathcal{P}$? Formally, we define a family of decision problems:
  Problem. r-TH-POWER-OF-$\mathcal{P}$-GRAPH
  Instance. A graph G.
  Question. Is $G = H^r$ for some graph $H \in \mathcal{P}$?
- *The r-th root problem*: Find some/all r-th roots of G which belong to $\mathcal{P}$.
- *The unique reconstruction problem*: Is the r-th root of G in $\mathcal{P}$ (if any) unique?

1998 ACM Subject Classification: G.2.2 Graph algorithms, F.2.2 Analysis of algorithms and problem complexity.

Key words and phrases: Graph roots, Graph powers, NP-completeness, Recognition algorithms.

Research supported by the Centre for Discrete Mathematics and its Applications (DIMAP), EPSRC award EP/D063191/1.

27th Symposium on Theoretical Aspects of Computer Science, Nancy, 2010
Editors: Jean-Yves Marion, Thomas Schwentick
Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.STACS.2010.2442

The above problems have been investigated for various graph classes $\mathcal{P}$. There exist characterizations of squares [15] and higher powers [3] of graphs, but they are not computationally efficient. Motwani and Sudan [14] proved the NP-completeness of recognizing graph squares and Lau [8] extended this to cubes of graphs. Motwani and Sudan [14] suggested that recognizing squares of bipartite graphs is also likely to be NP-complete. This was disproved by Lau [8], who gave a polynomial time algorithm that recognizes squares of bipartite graphs and counts the bipartite square roots of a given graph. Apparently the first proof that r-TH-POWER-OF-GRAPH and r-TH-POWER-OF-BIPARTITE-GRAPH are NP-complete for any $r \geq 3$ was recently announced in [10].

Considerable attention has been given to tree roots of graphs, which are quite well understood and can be computed efficiently, see Lin and Skiena [13], Kearney and Corneil [6] and Chang, Ko and Lu [2] who give a linear time algorithm for the r-th tree root of a given graph. Such a root need not be unique, not even up to isomorphism, so the difficulty lies in making consistent choices while constructing a root. Many techniques for computing tree roots rely on some sort of correspondence between vertex neighbourhoods in T and maximal cliques in T^p. We are going to use the computation of an r-th tree root of a graph as a black-box in our algorithms.

There has also been some work on the complexity of r-TH-POWER-OF-$\mathcal{P}$-GRAPH for such classes $\mathcal{P}$ as chordal graphs, split graphs and proper interval graphs [9] and for directed graphs and their powers [7].

In this work we address the above problems for another large family of graphs, namely graphs with no short cycles. Recall that the *girth* of a graph is the length of its shortest cycle (or ∞ for a tree). For convenience we shall denote by $\mathcal{GIRTH}_{\geq g}$ the class of all graphs of girth at least g, and by $\mathcal{GIRTH}^{+}_{\geq g}$ its subclass consisting of graphs with no vertices of degree one (which we call *leaves*). These classes of graphs make a convenient setting for graph roots because of the possible uniqueness results outlined below.

By [4] the recognition of squares of $\mathcal{GIRTH}_{\geq 4}$-graphs is NP-complete, while squares of $\mathcal{GIRTH}_{\geq 6}$-graphs can be recognized in polynomial time. The techniques of recognition (in this, and some other cases) include imposing some additional, local piece of information about the square root (like the existence of a certain edge) such that the root can then be reconstructed uniquely by expanding this data to the neighbouring vertices and eventually to the whole graph. Here we also exploit this idea.

For $r \geq 3$ no complexity-theoretic results have been known, but there is some very interesting work on the uniqueness of the roots. Precisely, Levenshtein et al. [12] proved that if G has a square root H in the class $\mathcal{GIRTH}^{+}_{\geq 7}$, then H is unique[1]. The same statement was extended in [11] to r-th roots in $\mathcal{GIRTH}^{+}_{\geq 2r+2\lceil (r-1)/4 \rceil +1}$, using a characterization of the neighbourhood of a vertex as the unique set satisfying a list of properties expressed in terms of the r-th power of the graph. The main conjecture in this area remains unresolved:

Conjecture 1.1 (Levenshtein, [11])**.** If a graph G has an r-th root H in $\mathcal{GIRTH}^{+}_{\geq 2r+3}$, then H is unique in that class.

The value of $g = 2r + 3$ is best possible, as witnessed by the cycle C_{2r+2}, which cannot be uniquely reconstructed from its r-th power. The best result towards Conjecture 1.1 is

[1]It is not possible to obtain uniqueness if the vertices of degree one are allowed, hence this technical restriction. See [12] for details.

that the number of roots H under consideration is at most $\delta(G)$ (the minimal vertex degree in G, [11]), but its proof yields only exponential time r-th root and recognition algorithms.

At the same time Farzad et al. made a conjecture about recognizing powers of graphs of lower-bounded girth:

Conjecture 1.2 (Farzad et al., [4])**.** The problem r-TH-POWER-OF-$\mathcal{GIRTH}_{\geq 3r-1}$-GRAPH can be solved in polynomial time.

Our contribution. Our first result gives an efficient reconstruction algorithm in Levenshtein's case:

Theorem 1.3. *Given any graph G, all its r-th roots in $\mathcal{GIRTH}^{+}_{\geq 2r+3}$ can be found in polynomial time.*

Next, we use this result to deal with the general case, i.e. when the roots are allowed to have leaves. It turns out that the same girth bound of $2r+3$ admits a positive result:

Theorem 1.4. *The problem r-TH-POWER-OF-$\mathcal{GIRTH}_{\geq 2r+3}$-GRAPH can be solved in polynomial time.*

Our result proves Conjecture 1.2 (for $r \geq 4$) and is in fact stronger. It also improves the result of [10] for $r = 3, g = 10$. Moreover, our algorithm for this problem is constructive and exhaustive in the sense that it finds "all" r-th roots in $\mathcal{GIRTH}_{\geq 2r+3}$ modulo the non-uniqueness of r-th tree roots of graphs, as explained in Section 4.

These positive results have a hardness counterpart:

Theorem 1.5. *The problem r-TH-POWER-OF-$\mathcal{GIRTH}_{\geq g}$-GRAPH is NP-complete for $g \leq r+1$ when r is odd and $g \leq r+2$ when r is even.*

The paper is structured as follows. First we prove some auxiliary results, useful both in the construction of algorithms and in the hardness result. Section 3 contains the main algorithm from Theorem 1.3, which is then used in Section 4 as a building block of the general recognition algorithm from Theorem 1.4. NP-completeness is proved in Section 5.

2. Auxiliary results

Let us fix some terminology. By $\mathrm{dist}_H(u, v)$ we denote the distance from u to v in H. The *d-neighbourhood* of a vertex u in H is the set of vertices of H which are exactly in distance d from u. The 1-neighbourhood (i.e. the set of vertices adjacent to u) will be denoted $N_H(u)$.

Our setup usually involves a pair of graphs G and H on a common vertex set V such that $G = H^r$. We adopt the notation

$$B_v := \{u \in V : \mathrm{dist}_H(u, v) \leq r\} = N_G(v) \cup \{v\}$$

for $v \in V$ (the letter B stands for "ball" of radius r in H). The lack of explicit reference to r and H in this notation should not lead to confusion. It is important that B_v depend only on G.

Almost all previous work on algorithmic aspects of graph powers [14, 4, 8, 9, 10] makes use of a special gadget, called *tail structure*, which, applied to a vertex u in G, ensures that in any r-th root H of G this vertex has the same, pre-determined neighbourhood. Our main observation is that in fact such a tail structure carries a lot more information about H. It pins down not just $N_H(u)$, but also each d-neighbourhood of u in H for $d = 1, \ldots, r$.

Lemma 2.1. *Let $G = H^r$ and suppose that $\{v_0, v_1, \ldots, v_r\} \subset V$ is a set of vertices such that $N_G(v_r) = \{v_{r-1}, \ldots, v_1, v_0\}$ and $N_G(v_{i+1}) \subset N_G(v_i)$ for all $i = 0, \ldots, r-1$, where the inclusions are strict.* [2]

Then the subgraph of H induced by $\{v_0, v_1, \ldots, v_r\}$ is a path $v_0 - v_1 - \ldots - v_r$ and the d-neighbourhood of v_0 in H is precisely

$$N_G(v_{r-d}) \setminus N_G(v_{r-d+1}) \cup \{v_d\}$$

for all $d = 1, \ldots, r$.

Proof. The subgraph K of H induced by $\{v_0, \ldots, v_r\}$ is connected — otherwise $N_G(v_r)$ would contain vertices from outside K. Consider any vertex u of K that has an edge to some vertex w outside K. Clearly, $\mathrm{dist}_K(v_r, u) = r$, since otherwise w would be in $N_G(v_r)$. This means that K is a path from v_r to u and u is the only vertex of that path which has edges to vertices outside K. The condition $N_G(v_{i+1}) \subset N_G(v_i)$ now implies that the vertices of this path are arranged as in the conclusion of the lemma. The second conclusion follows easily. ∎

Note that the tail structure itself does not enforce any extra constraints on H other than the d-neighbourhoods of v_0.

In the algorithm for r-TH-POWER-OF-$\mathcal{GIRTH}_{\geq 2r+3}$-GRAPH we will need to solve the following tree root problem with additional restrictions imposed on the d-neighbourhoods of a certain vertex:

Problem.	RESTRICTED-r-TH-TREE-ROOT
Instance.	A graph G, $r \geq 2$, a vertex $v \in V(G)$ and a partition $V(G) = \{v\} \cup T^{(1)} \cup \ldots \cup T^{(r)} \cup T^{(>r)}$.
Question.	Is $G = T^r$ for some tree T such that the d-neighbourhood of v in T is exactly $T^{(d)}$ for $d = 1, \ldots, r$?

Lemma 2.2. *There is a constructive polynomial time algorithm for RESTRICTED-r-TH-TREE-ROOT.*

Proof sketch. The neighbourhood-enforcing gadget from Lemma 2.1 can be attached to the given problem instance in such a way that the original graph has a restricted tree root if and only if the modified graph has any tree root (with no restrictions). Then the algorithms of [6, 2] apply to the modified instance. ∎

3. Algorithm for roots in $\mathcal{GIRTH}^+_{\geq 2r+3}$

In this section we present the algorithm from Theorem 1.3, that is the polynomial time reconstruction of all r-th roots in $\mathcal{GIRTH}^+_{\geq 2r+3}$ of a given graph G. There are two structural properties of graphs $H \in \mathcal{GIRTH}^+_{\geq 2r+3}$ that will be used freely throughout the proofs:

(*) Every $x \in V(H)$ is of degree at least 2 and the subgraph of H induced by B_x is a tree. This holds since any cycle in H within B_x would have length at most $2r+1$. We shall depict the ball B_x in H in the tree-like fashion.

[2] This assumption (strictness of inclusions) can be removed at the cost of a more complicated statement, but this generality is not needed here.

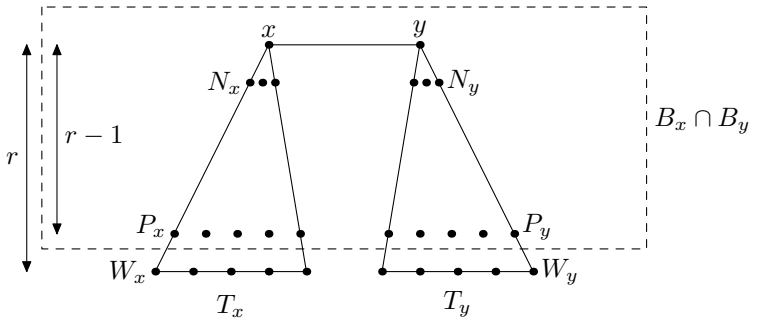

Figure 1: The subgraph of H induced by $B_x \cup B_y$.

(**) If there is a simple path from u to v in H of length exactly $r+1$ or $r+2$ then $u \notin B_v$. Indeed, $u \in B_v$ iff there is a path of length at most r from u to v in H, and combined with the first path this would yield a cycle of length at most $2r+2$.

To describe the algorithm we introduce the following sets:

$$S_{x,y} = B_x \cap B_y \setminus \bigcup_{v \in B_y \setminus B_x} B_v \setminus \{x\}$$

$$P_{x,y} = B_x \cap B_y \cap \bigcup_{v \in S_{x,y}} B_v$$

$$N_{x,y} = B_x \cap B_y \cap \bigcap_{v \in P_{x,y}} B_v \setminus \{x\}$$

Defined for arbitrary $x, y \in V$, these sets are probably quite meaningless for the reader. The definitions are motivated by the proof of the next theorem, in which we determine these sets in more familiar terms for the endpoints x, y of an actual edge in some r-th root of G. Precisely:

Theorem 3.1. *Suppose $G = H^r$ for a graph $H \in \mathcal{GIRTH}^+_{\geq 2r+3}$ and $xy \in E(H)$. Then*

$$N_{x,y} = N_H(x).$$

Proof. Because of the girth condition the set $B_x \cup B_y$ in H consists of two disjoint trees T_x and T_y, rooted in x and y respectively and connected by the edge xy (see Fig.1). Let us introduce some subsets of those trees. By W_x and W_y denote the last levels:

$$W_x = \{u \in T_x : \mathrm{dist}_H(u,x) = r\}, \quad W_y = \{u \in T_y : \mathrm{dist}_H(u,y) = r\},$$

by P_x and P_y the next-to-last levels:

$$P_x = \{u \in T_x : \mathrm{dist}_H(u,x) = r-1\}, \quad P_y = \{u \in T_y : \mathrm{dist}_H(u,y) = r-1\},$$

and by N_x and N_y the children of x and y in T_x and T_y:

$$N_x = \{u \in T_x : \mathrm{dist}_H(u,x) = 1\}, \quad N_y = \{u \in T_y : \mathrm{dist}_H(u,y) = 1\}.$$

Clearly $B_x \cap B_y = (T_x \setminus W_x) \cup (T_y \setminus W_y)$, $W_x = B_x \setminus B_y$ and $W_y = B_y \setminus B_x$. Note that if $r = 2$ we have $N_x = P_x$ and $N_y = P_y$.

First observe that every $u \in N_x$ and every $v \in B_y \setminus B_x = W_y$ are connected by a path of length $r+2$. It follows by (**) that $u \notin B_v$, which implies

$$N_x \subset S_{x,y}.$$

It is also clear that $S_{x,y} \subset T_x$ (because every vertex in T_y has a descendant $v \in W_y$).

Now the sum $\bigcup_{v \in S_{x,y}} B_x \cap B_y \cap B_v$ contains $\bigcup_{v \in N_x} B_x \cap B_y \cap B_v = (B_x \cap B_y) \setminus P_y$. On the other hand, if $v \in S_{x,y}$ and $u \in P_y$ then $u \notin B_v$. Indeed, if $u \in B_v$ then there would be a path from u to v of length at most r. This path cannot be contained in $T_x \cup T_y$ (because $\mathrm{dist}_H(u,x) = r$, so one can only get as far as x going from u), hence it must exit T_y through W_y and then enter T_x through W_x, finally reaching $v \in S_{x,y}$. However, that yields a path from W_y to $S_{x,y}$ of length at most r (in fact at most $r-1$), contradicting the definition of $S_{x,y}$. Eventually we proved

$$P_{x,y} = (B_x \cap B_y) \setminus P_y.$$

Now we have $\{y\} \cup N_x \subset N_{x,y}$ because every vertex of $\{y\} \cup N_x$ is in distance at most r from all the vertices of $(B_x \cap B_y) \setminus P_y$. On the other hand, for every vertex u of $B_x \cap B_y$ that is not in $N_x \cup \{x,y\}$ one can find a path of length $r+1$ that starts in u and ends in a vertex $v \in (B_x \cap B_y) \setminus P_y$. Then, according to (**), $u \notin B_v$, so $u \notin N_{x,y}$. Such a path is obtained by going from u up the tree it is contained in (T_x or T_y) and then down in the other tree.

Concluding, we have identified $N_{x,y}$ to be $N_x \cup \{y\}$, as required. ∎

The previous theorem should be understood as follows. Given a graph G, we want to find its r-th root H. If we fix at least one edge xy of H in advance, we can compute the neighbourhood $N_H(x)$ of x using only the data available in G. But then we can move on in the same way, computing the neighbours of those neighbours etc.

Algorithm 1 Input: G,r. **Output:** All r-th roots of G in $\mathcal{GIRTH}^{+}_{\geq 2r+3}$

pick a vertex x with smallest $|B_x|$
for all y in B_x **do**
  H =reconstructFromOneEdge(G, xy)
  if $H \in \mathcal{GIRTH}^{+}_{\geq 2r+3}$ and $H^r = G$ **output** H
end for

reconstructFromOneEdge(G, e):
$H = (V(G), \{e\})$
for all $u \in V$ **set** processed[u]:=**false**
while H has an unprocessed vertex x of degree at least 1 **do**
  y = any neighbour of x in H
  $E(H) = E(H) \cup \{xz$ for all $z \in N_{x,y}\}$
  processed[x]:=**true**
end while
return H

The r-th root algorithm is now straightforward. The procedure *reconstructFromOneEdge* attempts to compute H from G assuming the existence of a given edge e in H. This is repeated for all possible edges from a fixed vertex x. It remains to notice that $N_{x,y}$ can be computed in polynomial time.

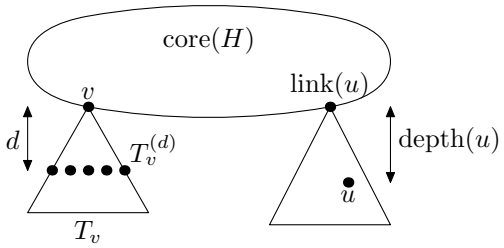

Figure 2: The notation of Section 4.

4. Removing the no-leaves restriction

In this section we obtain a polynomial time algorithm for the general recognition problem r-TH-POWER-OF-$\mathcal{GIRTH}_{\geq 2r+3}$-GRAPH, proving Theorem 1.4. We start with a few definitions (see Fig.2).

For a graph H, which is not a tree, let core(H) denote the largest induced subgraph of H whose every vertex has degree at least two. Alternatively this can be defined as follows. Given H, let H' be the graph obtained from H by removing all *leaves* (vertices of degree one) and inductively define $H^{(1)} = H'$, $H^{(n)} = (H^{(n-1)})'$. This process eventually stabilizes at the graph core(H).

A vertex $v \in V(H)$ is called a *core vertex* if it belongs to core(H) and a *non-core vertex* otherwise. The non-core vertices are grouped into trees attached to the core. For every vertex $v \in$ core(H) we denote by T_v the tree attached at v (including v) and by $T_v^{(d)}$ (for $d \geq 0$) the set of vertices of T_v located in distance d from v. For a non-core vertex u the *link* of u (denoted link(u)) is its closest core vertex and the *depth* of u (denoted depth(u)) is the distance from u to link(u).

4.1. Outline of the algorithm.

The algorithm for r-TH-POWER-OF-$\mathcal{GIRTH}_{\geq 2r+3}$-GRAPH processes the input graph G in several steps (see Algorithm 2). First, we check if G has a tree r-th root [6, 2]. If not, then we split the vertices of G into the core and non-core vertices of any of its r-th roots. Lemma 4.1 shows how to find such a partition and ensures that it is uniquely determined only by the graph G.

Let $\tilde{G}$ be the subgraph of G induced by all the vertices that are classified as belonging to the core of any possible r-th root H. We now employ the algorithm from the previous section to find all r-th roots $\tilde{H}$ of $\tilde{G}$ which have girth at least $2r + 3$ and no leaves (there are at most $\delta(G)$ of them; conjecturally there is at most one).

Finally, we must attach the non-core vertices to each of the possible $\tilde{H}$. It turns out that once the core is fixed, the link of each non-core vertex can be uniquely determined, so we can pin down all the sets $V(T_v)$. However, we cannot simply look for any r-th tree root of the subgraph of G induced by $V(T_v)$, because we have to ensure that the tree structure that we are going to impose on $V(T_v)$ is compatible with the neighbourhood information contained in the rest of G. Fortunately Lemma 4.2 guarantees that for a fixed G and core(H), all the sets $T_v^{(d)}$ for $d = 1, \ldots, r$ are also uniquely determined. Since all the distances from the vertices of T_v to the rest of the graph depend only on the vertex depths and the structure of the core, this is exactly the additional piece of data we need. Any tree root satisfying

the given depth constraints will be compatible with the rest of the graph. Concluding, the problem we are left with for each T_v is the RESTRICTED-r-TH-TREE-ROOT from Section 2. If all these instances have positive solutions, then the graph H defined as $\tilde{H}$ with the trees T_v attached at each core vertex v is an r-th root of G.

The next two subsections describe the two crucial steps: detecting non-core vertices and the reconstruction of trees T_v.

4.2. Finding core and non-core vertices.

The next lemma shows how to detect all vertices located "close to the bottom" of the trees T_v in H.

Lemma 4.1. *Suppose $H \in \mathcal{GIRTH}_{\geq 2r+3}$ and $H^r = G$. Then the following conditions are equivalent for a vertex $u \in H$:*

(1) *$u \notin H^{(r)}$.*
(2) *There is some vertex $v \in H$, $v \neq u$ such that $B_u \subseteq B_v$.*

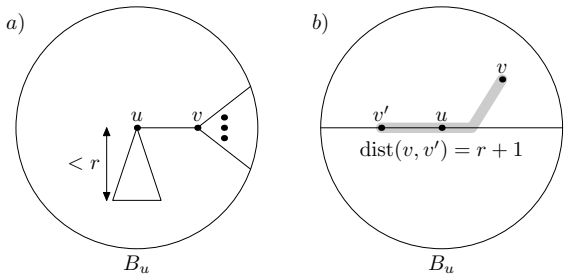

Figure 3: The proof of Lemma 4.1.

Proof. If $u \notin H^{(r)}$, then by the definition u becomes a leaf after at most $r-1$ steps of the leaf-removal procedure and is removed in the subsequent step. Let v be the last vertex adjacent to u just before u is removed (see Fig.3a). Clearly all the vertices reachable from u in at most r steps are also reachable from v in at most r steps, so $B_u \subseteq B_v$.

If, on the other hand, $u \in H^{(r)}$ then u is not removed in the first r steps of cutting off the leaves of H, which means there exist at least two disjoint paths of length r starting at u (see Fig.3b). However, it implies that for every vertex $v \in B_u$ there exists another $v' \in B_u$ (on one of those paths) such that $\mathrm{dist}_H(v, v') = r+1$, hence $v' \in B_u \setminus B_v$. Therefore B_u is not contained in B_v for any $v \neq u$. ∎

Recursively deleting all vertices u such that $B_u \subseteq B_v$ for some $v \neq u$ determines the consecutive sets $V(H^{(r)})$, $V(H^{(2r)})$, $V(H^{(3r)})$, ... for *any* r-th root $H \in \mathcal{GIRTH}_{\geq 2r+3}$ of G using only the information available in G. Eventually we obtain $V(\mathrm{core}(H))$ which is the vertex set of $\tilde{G}$.

4.3. Attaching the trees T_v.

For each possible $\mathrm{core}(H)$ we need to decide on a way of attaching the remaining (non-core) vertices to H in a way which ensures that $H^r = G$. It turns out that all the data necessary to ensure the compatibility can be read off from G and $\mathrm{core}(H)$, so again this data is common for all the possible r-th roots of G that have a fixed core.

Lemma 4.2. *Suppose that $H \in \mathcal{GIRTH}_{\geq 2r+3}$ is a graph such that H is not a tree and $H^r = G$. Then for every non-core vertex u of H we have:*

- *either $B_u \cap V(core(H)) = \emptyset$, in which case depth$(u) > r$, or*
- *the subgraph of H induced by $B_u \cap V(core(H))$ is a tree whose only center is link(u) and whose height (the distance from the center to every leaf) is $r -$ depth(u).*

Proof. The first statement is obvious. As for the second, the subgraph induced by $B_u \cap V(\mathrm{core}(H))$ consists of all the vertices of $V(\mathrm{core}(H))$ in distance at most $r - \mathrm{depth}(u)$ from $\mathrm{link}(u)$. Since $\mathrm{core}(H)$ is a graph of girth at least $2r+3$ with no degree one nodes, these vertices induce a tree in H, and all the leaves of this tree are exactly in distance $r - \mathrm{depth}(u)$ from $\mathrm{link}(u)$. Therefore $\mathrm{link}(u)$ is the unique center of that tree. ∎

Lemma 4.2 yields a method of partitioning the non-core vertices into the sets $V(T_v)$ and subdividing each $V(T_v)$ into a disjoint union $\{v\} \cup T_v^{(1)} \cup \ldots \cup T_v^{(r)} \cup T_v^{(>r)}$ of vertices in distance $1, 2, \ldots, r$ and more than r from v using only the data from G and $\mathrm{core}(H)$. Indeed, for the vertices u with $B_u \cap V(\mathrm{core}(H)) \neq \emptyset$ one finds the center and height of the subtree of $\mathrm{core}(H)$ induced by $B_u \cap V(\mathrm{core}(H))$ and applies the second part of Lemma 4.2 to obtain both $\mathrm{link}(u)$ and $\mathrm{depth}(u)$, thus classifying u to the appropriate $T_v^{(d)}$. The links of all remaining vertices are determined using the fact that all vertices in one connected component of $G \setminus \bigcup_{v \in \mathrm{core}(H), d=0,\ldots,r-1} T_v^{(d)}$ have the same link.

Algorithm 2
Input: G,r.
Output: r-th roots of G in $\mathcal{GIRTH}_{\geq 2r+3}$ (one per each core)

check if $G = T^r$ for some tree T

$\tilde{G} := G$
while $\tilde{G}$ has vertices u, v with $B_u \subseteq B_v$ **do**
  remove from $\tilde{G}$ all u such that $B_u \subseteq B_v$ for some v
end while

for every graph $\tilde{H} \in \mathcal{GIRTH}^+_{\geq 2r+3}$ such that $\tilde{H}^r = \tilde{G}$ **do**
  $H := \tilde{H}$
  for every vertex $v \in V(\tilde{H})$ **do**
    find $V(T_v)$ and a partition $V(T_v) = \{v\} \cup T_v^{(1)} \cup \ldots \cup T_v^{(r)} \cup T_v^{(>r)}$
    use *restrictedTreeRoot* to reconstruct some tree T_v
    extend H by attaching T_v at v
  end for
  if all T_v existed **output** H
end for

5. Hardness results

Now we sketch the hardness of recognition for powers of graphs of lower-bounded girth (Theorem 1.5). For the reductions we use the following NP-complete problem (see [5, Prob. SP4]). It has already been successfully applied in this context ([4, 8, 9, 10]).

Problem. HYPERGRAPH 2-COLORABILITY (H2C)
Instance. A finite set S and a collection $S_1, \ldots, S_m$ of subsets of S.
Question. Can the elements of S be colored with two colors A, B such that each set S_j has elements of both colors?

An instance of this problem (also known as SET-SPLITTING) will be denoted $\mathcal{S} = (S; S_1, \ldots, S_m)$. We shall refer to the elements of the universum S as $x_1, \ldots, x_n$. Any assignment of colors A and B to the elements of S which satisfies the requirements of the problem will be called a *2-coloring*.

In this section we fix r and let $k = \lfloor \frac{r}{2} \rfloor$, so that $r = 2k$ or $r = 2k + 1$ depending on parity.

5.1. Case of odd $r = 2k + 1$

Consider an instance $\mathcal{S} = (S; S_1, \ldots, S_m)$ of H2C. The following two definitions describe an auxiliary graph that will be used as a base for further constructions. The reader is referred to Fig.4 for a self-explanatory presentation of the graphs $K_\mathcal{S}$ and $H_\mathcal{S}$ defined below.

Definition 5.1. For an instance $\mathcal{S} = (S; S_1, \ldots, S_m)$ let $V_\mathcal{S}$ be the following set of vertices:

- S_j, x_i for all subsets and elements,
- A, B, X,
- $T_{i,j}^{(l)}$ for every pair i, j such that $x_i \in S_j$ and every $l = 1, \ldots, k-1$,
- $P_i^{(l)}$ for every x_i and every $l = 1, \ldots, k-1$,
- the tail vertices $S_j^{(l)}$ for each j and $l = 1, \ldots, r$.

Definition 5.2. Given any instance $\mathcal{S} = (S; S_1, \ldots, S_m)$ define a graph $K_\mathcal{S}$ on the vertex set $V_\mathcal{S}$ with the following edges:

- a path $S_j - T_{i,j}^{(1)} - \ldots - T_{i,j}^{(k-1)} - x_i$ whenever $x_i \in S_j$,
- a path $x_i - P_i^{(1)} - \ldots - P_i^{(k-1)}$ for every x_i,
- $X - x_i$ for all i,
- the tail paths, that is $S_j - S_j^{(1)} - S_j^{(2)} - \ldots - S_j^{(r)}$ for every j.

This graph encodes only the structure of $\mathcal{S}$. To encode the coloring we link the loose paths from x_i to either A or B.

Definition 5.3. Given an instance $\mathcal{S}$ and a color assignment, define the graph $H_\mathcal{S}$ to be $K_\mathcal{S}$ with the additional edges $P_i^{(k-1)} - A$ whenever x_i has color A and $P_i^{(k-1)} - B$ whenever x_i has color B.

Note that $H_\mathcal{S}$ has girth $2k + 2 = r + 1$. Now comes the graph to be used in our NP-completeness reduction:

Definition 5.4. For any instance $\mathcal{S} = (S; S_1, \ldots, S_m)$ of H2C put

$$G_\mathcal{S} = {K_\mathcal{S}}^r \cup E_\mathcal{S}$$

where $E_\mathcal{S}$ is the set of edges from A and B to each of X, x_i, S_j, $T_{i,j}^{(l)}$, $P_i^{(l)}$, and $S_j^{(1)}$ for all possible i, j, l.

Observe that $G_\mathcal{S}$ is defined independently of any particular color assignment. Moreover, by analyzing Fig.4 it is not hard to check the following lemma:

Lemma 5.5. *For any 2-colored instance $\mathcal{S}$ we have $G_\mathcal{S} = {H_\mathcal{S}}^r$.* ∎

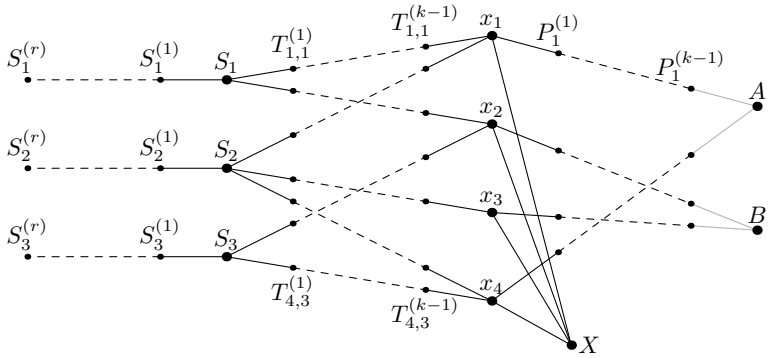

Figure 4: For $\mathcal{S} = (\{x_1, \ldots, x_4\}; \{x_1, x_2\}, \{x_1, x_3, x_4\}, \{x_2, x_4\}\})$ the graph $K_{\mathcal{S}}$ consists of all *but* the shaded edges. The graph $H_{\mathcal{S}}$ (made of all the edges above) encodes the coloring with x_1, x_4 of color A and x_2, x_3 of color B. It is a 2-coloring of $\mathcal{S}$ since all S_j are in distance $2k$ from A and B.

Proof of Theorem 1.5 for odd r. Given an instance $\mathcal{S} = (S; S_1, \ldots, S_m)$ construct the graph $G_{\mathcal{S}}$. If $\mathcal{S}$ has a 2-coloring, then $G_{\mathcal{S}}$ is the r-th power of a graph with girth at least $r+1$, namely $G_{\mathcal{S}} = H_{\mathcal{S}}{}^r$ by Lemma 5.5.

For the inverse implication suppose that $G_{\mathcal{S}} = H^r$ for some graph H. Define the coloring as follows: x_i has color A (resp. B) if there is a path of length at most k from x_i to A (resp. B) in H. Clearly each x_i is assigned at most one color since otherwise A and B would be adjacent in H^r.

The tail structure $S_j, S_j^{(1)}, \ldots, S_j^{(r)}$ of each S_j satisfies the assumptions of Lemma 2.1, so it enforces that in H:

- for every j the k-neighbourhood of S_j is precisely $\{x_i : x_i \in S_j\} \cup \{S_j^{(k)}\}$ (as in $K_{\mathcal{S}}$),
- A and B are exactly in distance $2k$ from each S_j (by the definition of $E_{\mathcal{S}}$).

Therefore for each j there has to be at least one vertex in $\{x_i : x_i \in S_j\}$ that is k steps from A and at least one that is k steps from B. This proves that the obtained coloring solves the H2C instance. ∎

5.2. Case of even $r = 2k$

We omit this case for reasons of space. The argument is similar, but requires a slight modification to the graphs $K_{\mathcal{S}}$, $H_{\mathcal{S}}$ and $G_{\mathcal{S}}$.

6. Conclusions and open problems

In this work we presented an efficient algorithmic solution to Levenshtein's reconstruction conjecture and we applied it to a more general, unrestricted r-th root problem. From a high-level perspective, it was possible because we could extract the "core of the problem" which has very few solutions (as the conjecture suggests), so we could hope that these can be found quickly. We also hope that the reverse flow of ideas is possible, so that some improved algorithmic edge-by-edge reconstruction technique might help resolve Levenshtein's conjecture.

Another (probably challenging) problem is to find a complete girth-parametrized complexity dichotomy, that is to close the gap between $r+1$ (or $r+2$) and $2r+3$. We believe that the r-th power recognition remains NP-complete even for graphs of girth $2r$.

In fact it would even be very interesting to investigate possible complexity results for finding square roots in $\mathcal{GIRTH}_{\geq 5}$ or $\mathcal{GIRTH}^{+}_{\geq 5}$ (completing the complexity dichotomy of [4]). Note that the complete graph $G = K_n$ has a square root in the class $\mathcal{GIRTH}^{+}_{\geq 5}$ if and only if there exists a graph on n vertices that has girth 5 and diameter 2. By the Hoffman-Singleton theorem (see [16, 1]) such a graph may exist only for $n = 5, 10, 50$ and 3250. The first three of these graphs are known, and the existence of the last one (for $n = 3250$) is a long-standing open problem. Therefore, any efficient algorithm for SQUARE-OF-$\mathcal{GIRTH}^{+}_{\geq 5}$-GRAPH might (at least in principle) solve this problem.

Acknowledgement

The authors thank the anonymous STACS referees for helpful comments.

References

[1] N.Biggs, Algebraic Graph Theory, Cambridge Univ. Press
[2] Maw-Shang Chang, Ming-Tat Ko, Hsueh-I Lu, *Linear-Time Algorithms for Tree Root Problems*, Proc. 10th SWAT, LNCS 4059 (2006)
[3] F.Escalante, L.Montejano, T.Rojano, *Characterization of n-path graphs and of graphs having nth root*, Journal of Combinatorial Theory, Series B, 16: 282-298 (1974)
[4] Babak Farzad, Lap Chi Lau, Van Bang Le, Nguyen Ngoc Tuy, *Computing Graph Roots Without Short Cycles*, Proc. 26th STACS (2009) 397-408
[5] M.R.Garey, D.S.Johnson, *Computers and Intractability — A Guide to the Theory of NP-Completeness*, Freeman, Oxford, UK, 1979
[6] P.E.Kearney, D.G.Corneil *Tree powers*, Journal of Algorithms 29 (1998) 111-131
[7] Martin Kutz, *The complexity of Boolean matrix root computation*, Theor. Comp. Sci. 325 (2004) 373-390
[8] Lap Chi Lau, *Bipartite Roots of Graphs*, ACM Transactions on Algorithms, Vol.2, No.2, April 2006, 178-208
[9] Lap Chi Lau, Derek G. Corneil *Recognizing Powers of Proper Interval, Split and Chordal Graphs*, SIAM J. Discrete Math., Vol.18, No.1, 2004, 83-102
[10] Van Bang Le and Ngoc Tuy Nguyen, *Hardness Results and Efficient Algorithms for Graph Powers*, WG 2009
[11] V.I. Levenshtein, *A conjecture on the reconstruction of graphs from metric balls of their vertices*, Discrete Mathematics 308(5-6): 993-998 (2008)
[12] V.I. Levenshtein, E.V. Konstantinova, E.Konstantinov, S.Molodtsov, *Reconstruction of a graph from 2-vicinities of its vertices*, Discrete Applied Mathematics 156(9): 1399-1406 (2008)
[13] Y.-L.Lin, S.S.Skiena, *Algorithms for square roots of graphs*, SIAM J. Discrete Math. 8 (1995), 99-118
[14] R.Motwani, M.Sudan, *Computing Roots of Graphs is Hard*, Discrete Applied Mathematics 54(1): 81-88 (1994)
[15] A.Mukhopadhyay, *The square root of a graph*, Journal of Combinatorial Theory, Series B, 2: 290-295 (1967)
[16] R.R.Singleton, *There is no irregular Moore graph*, American Mathematical Monthly 75, vol 1 (1968) 42-43