Matemática
Contemporânea
Vol. 55, 133–141
http://doi.org/10.21711/231766362023/rmc5515

Faster computing of graph square roots with girth at least six

Cristopher Carcereri, **Aleffer Rocha** and **Renato Carmo**

Federal University of Paraná, R. Coronel Francisco Heráclito dos Santos 100, 81531-980, Curitiba, PR, Brazil

Dedicated to Professor Jayme Szwarcfiter on the occasion of his 80th birthday

Abstract. We consider the problem of finding a graph which is a square root of girth at least k of a graph G with n vertices and m edges, for $k \in \{6, 7\}$. The best-known solutions for these problems are an $\mathcal{O}(\delta(G) \cdot n^4)$ algorithm for $k = 6$ and an $\mathcal{O}(m \cdot n^2)$ algorithm for $k = 7$. We show that it is possible to solve these problems in time $\mathcal{O}(\delta(G) \cdot n^2)$ for $k = 6$ and $\mathcal{O}(n^2)$ for $k = 7$.
Keywords: graph square roots, cycle detection, algorithm complexity

2020 Mathematics Subject Classification: 05C76, 68R10, 68W40.

1 Introduction

The *square* of a graph H is the graph H^2 obtained by adding to H edges joining all vertices at distance 2. We say that H is a *square root* of

Corresponding author: `arocha@inf.ufpr.br`

G if $G = H^2$. Not every graph has a square root. On the other hand, a graph can have several square roots.

The problem of deciding if a given graph has a square root is $\mathcal{NP}$-complete [5]. The problem of computing a square root of a given graph is, therefore, $\mathcal{NP}$-hard.

Given a graph class $\mathcal{C}$, a related and relevant problem is, given a graph G, computing a square root of G belonging to $\mathcal{C}$. This problem is called the $\mathcal{C}$-square root problem.

Let $\mathcal{G}_k$ denote the class of graphs of girth at least k. An interesting dichotomy exists with respect to the $\mathcal{G}_k$-square root problems, namely, the $\mathcal{G}_k$-square root problem is polynomially solvable if $k \geq 6$ and is $\mathcal{NP}$-hard otherwise [3–5].

Also, if there is a square root in $\mathcal{G}_6$, it is unique up to isomorphism [1].

That $\mathcal{G}_k$-square root is polynomially solvable for $k \geq 6$ was proved in [3]. In doing so, the authors introduce an $\mathcal{O}(\delta(G) \cdot n^4)$ algorithm for $\mathcal{G}_6$-square root (where $\delta(G)$ denotes the minimum degree in G) and an $\mathcal{O}(m \cdot n^2)$ algorithm for $\mathcal{G}_7$-square root. Here we improve these algorithms showing that $\mathcal{G}_6$-square root can be solved in time $\mathcal{O}(\delta(G) \cdot n^2)$ and that $\mathcal{G}_7$-square root can be solved in time $\mathcal{O}(n^2)$.

The text is organized as follows. Section 1.1 introduces some definitions and the notation used. Section 2 discusses the algorithm of [3] for $\mathcal{G}_6$-square root. Section 3 explains the modification proposed to the algorithm described in Section 2 and performs the correspondent analysis. Section 4 discusses our $\mathcal{O}(n^2)$ time algorithm for the $\mathcal{G}_7$-square root problem.

1.1 Definitions and notation

A (simple) *graph* is a pair $G = (V(G), E(G))$ where $V(G)$ is a finite set and $E(G) \subseteq \binom{V(G)}{2}$. Their elements are called *vertices* and *edges* of G, respectively. We follow the standard definitions for graph related concepts. As usual, we denote an edge $\{u, v\}$ by uv whenever possible. If v is a vertex of G, we denote its neighborhood in G by $N_G(v)$ and its closed neighborhood in G (that is $N_G(v) \cup \{v\}$) by $N_G[v]$. The distance between vertices u and v in G is denoted $d_G(u,v)$. The minimum degree of G is denoted $\delta(G)$. Cycles of length n are denoted by C_n. The square of a graph G is the graph G^2 where $V(G^2) = V(G)$ and $E(G^2) = \{uv\colon d_G(u,v) \leq 2\}$. A square root of G is a graph whose square is G.

As in Section 1, for each $k \geq 3$ we denote the class of graphs of girth at least k by $\mathcal{G}_k$ and define the $\mathcal{G}_k$-square root problem as the problem of, given a graph G, compute a square root of G belonging to $\mathcal{G}_k$ or determining that no such root exists.

2 Square roots with girth at least 6

Farzad et al. [3] show that it is possible to find a square root of girth at least 6 of a given graph or to determine that no such root exists in polynomial time. Their algorithm corresponds to the G_6-Sqrt(G) procedure.

G_6-Sqrt(G)

Input: a connected graph G with at least 3 vertices

Output: a square root of G with girth at least 6, if it exists; "Does Not Compute", otherwise

$v \leftarrow$ a minimum degree vertex of G

For *each* $u \in N_G(v)$

  $H \leftarrow G_6$-SqrtEdge(G, uv)

  If $H \neq$ "Does Not Compute"

    Return H

Return "Does Not Compute"

Algorithm G_6-Sqrt(G) and the following discussion assume that G is connected and has at least 3 vertices. The square of a graph is the union of the squares of its connected components, and every connected graph with less than 3 vertices is a square root of itself.

We refer the reader to [3] for a full discussion of the correctness of algorithm G_6-Sqrt and limit ourselves to state the propositions upon which said correctness is based plus some brief comments.

Proposition 2.1 (Lemma 3.1 in [3]). *Let H be a connected $\{C_3, C_5\}$-free graph and let $G = H^2$. For all $v \in V(H)$ and all $u \in N_H(v)$,*

$$N_H(u) = N_G(u) \cap (N_G[v] - N_H(v)).$$

Proposition 2.2 (Lemma 3.3 in [3]). *Let H be a graph of girth at least 6, let $uv \in E(H)$ and let $G = H^2$. The graph $G[N_G(u) \cap N_G(v)]$ has at most 2 connected components. Moreover, if A and B are the vertex sets of these components (one of them may be empty), then (i) $A = N_H(u) - \{v\}$ and $B = N_H(v) - \{u\}$, or (ii) $B = N_H(u) - \{v\}$ and $A = N_H(v) - \{u\}$.*

```
G_6-SqrtEdge(G, uv)
  Input: a connected graph G with at least 3 vertices and an edge uv of G
  Output: a square root H of G with girth at least 6 such that
          uv ∈ E(H), if it exists; "DOES NOT COMPUTE", otherwise
  K ← G[N_G(u) ∩ N_G(v)]
  If K has one or two components
     A ← the vertex set of a (non-empty) component of K
     H ← G_6-SqrtNgbh(G, v, A ∪ {u})
     If H ≠ "DOES NOT COMPUTE"
        Return H
     Return G_6-SqrtNgbh(G, u, A ∪ {v})
  Return "DOES NOT COMPUTE"
```

Suppose H is a square root of G with girth at least 6. Proposition 2.2 tells us that if $uv \in E(H)$ and A is the vertex set of a component of $G[N_G(u) \cap N_G(v)]$, then[1] either (i) $N_H(u) = A \cup \{v\}$ or (ii) $N_H(v) = A \cup \{u\}$. Besides, if the neighborhood in H of a vertex $x \in V(G)$ is known, Proposition 2.1 tells us how to compute the neighborhood in H of every vertex in $N_H(x)$. Besides, if the neighborhood in H of a vertex $x \in V(G)$ is known, Proposition 2.1 tells us how to compute the neighborhood in H of every vertex in $N_H(x)$.

Algorithm G_6-Sqrt(G) chooses a minimum degree vertex $v \in V(G)$ and, for each $u \in N_G(v)$, calls G_6-SqrtEdge(G, uv) trying to find a root of girth at least 6 of G containing this edge. Algorithm G_6-SqrtEdge(G, uv) uses Proposition 2.2 to determine the possible neighborhood of u and v in

[1]The algorithm in [3] also considers the possibilities that (i) $N_H(v) = B \cup \{u\}$ or (ii) $N_H(u) = B \cup \{u\}$. Note however that, by symmetry, we can consider either pair of conditions without loss of generality.

this root, and calls G_6-SqrtNgbh for both cases. G_6-SqrtNgbh(G, v, U) uses a BFS-like procedure that computes H if $N_H(v) = U$. As $N_H(v) \neq U$ may be the case, we need to check if the $\{C_3, C_5\}$-free output by the algorithm is indeed a root of G. Also, to guarantee it has girth at least 6, we need to check if it is C_4-free. Algorithm Check(G, H) tests these conditions: as H is $\{C_3, C_5\}$-free, if it is also C_4-free and $H^2 = G$, then it is a solution.

```
G6-SqrtNgbh(G, v, U)
---------------------------------------------------------------
Input: a connected graph G with at least 3 vertices, a vertex v of G and
       a nonempty set U ⊆ N_G(v)
Output: a square root H of G with girth at least 6 such that
        N_H(v) = U, if it exists; "DOES NOT COMPUTE", otherwise
Q ← empty queue
H ← empty graph
For each u ∈ V(G)
    u.parent ← NULL
For each u ∈ U
    add uv to H
    add u to Q
    u.parent ← v
While Q is not empty
    remove a vertex u from Q
    X ← N_G[u.parent] − N_H(u.parent)
    W ← N_G(u) ∩ X
    For each w ∈ W
        add uw to H
        If w.parent = NULL
            add w to Q
            w.parent ← u
Return Check(G, H)
```

The analysis in [3] concludes that if $n = |V(G)|$, then algorithm G_6-Sqrt(G) runs in time $\mathcal{O}(\delta(G) \cdot n^4)$, where the $\mathcal{O}(n^4)$ term comes from the time needed for testing if H has a C_4 in algorithm Check(G, H). Moreover, their analysis considers that testing if $H^2 = G$ has the time complexity of multiplying two $n \times n$ matrices ($\mathcal{O}(n^{2.373})$ as of today [2]).

We show in Section 3 that it is possible to combine the test if a n-vertex graph is C_4-free *and* the test if $H^2 = G$ in a single-time $\mathcal{O}(n^2)$ algorithm. The next result of these improvements is that computing a square root of girth at least 6 of an n-vertex graph G can be done in $\mathcal{O}(\delta(G) \cdot n^2)$ time.

Check(G, H)

Input: a graph G and a $\{C_3, C_5\}$-free graph H

Output: H, if H is C_4-free and a square root of G; "DOES NOT COMPUTE", otherwise

```
If H has a C4
    Return "DOES NOT COMPUTE"
If H^2 ≠ G
    Return "DOES NOT COMPUTE"
Return H
```

---

3 Checking a solution

The following algorithm decides if a given n-vertex graph is C_4-free in time $\mathcal{O}(n^2)$.

C_4-free(H)

Input: a graph H

Output: **yes**, if H is C_4-free; **no**, otherwise

```
M ← a 0-initialized matrix indexed by V(H) × V(H)
For each v ∈ V(H)
    For each uw ∈ (N_H(v) choose 2)
        If M[u, w] = 1
            Return no
        M[u, w] ← M[w, u] ← 1
Return yes
```

Algorithm C_4-free(H) is a sort of "folklore algorithm" (see, for example, [6]). Its idea is very simple: $M[u, v]$ counts the number of common neighbors of vertices u and v. If the count exceeds 1, then there is a C_4 formed by u, v and the common neighbors and the algorithm returns **no**.

On the other hand, if the algorithm returns **yes** and H is also C_3-free, then the matrix M computed by algorithm C_4-free(H) is such that

$M[u,v] = 1$ if and only if $d_H(u,v) = 2$. In this case, the sum of M with the adjacency matrix H results in the adjacency matrix of H^2.

We can substitute algorithm $\mathsf{Check}(G,H)$ by the following algorithm.

```
ImprovedCheck(G, H)
---------------------------------------------------------------
Input: a graph G and a {C_3, C_5}-free graph H
Output: H, if H is C_4-free and a square root of G; "DOES NOT
        COMPUTE", otherwise
M ← adjacency matrix of H
For each v ∈ V(G)
    For each uw ∈ (N_H(v) choose 2)
        If M[u,w] = 1
            Return "DOES NOT COMPUTE"
        M[u,w] ← M[w,u] ← 1
If M is the adjacency matrix of G
    Return H
Return "DOES NOT COMPUTE"
```

4 Square roots with girth at least 7

In this section we introduce an $\mathcal{O}(n^2)$ algorithm for the $\mathcal{G}_7$-square root problem. The algorithm is based on the following statement.

Proposition 4.1. *Let H be a graph of girth at least 7 and let $G = H^2$. If $uv \in E(G)$ but $uv \notin E(H)$, then u and v have only one common neighbor w in H and $N_G[u] \cap N_G[v] = N_H[w]$.*

Proof. Let H, G, u and v be as above. As $uv \in E(G) - E(H)$, there must be a neighbor w common to u and v in H. Besides, no other such common neighbor can exist or H would have a C_4 and its girth would not be 7. Every vertex in $N_H[w]$ has distance at most 2 from u and v in H, thus $N_H[w] \subseteq N_G[u] \cap N_G[v]$. If there was a vertex a in $(N_G[u] \cap N_G[v]) - N_H[w]$, there would be a cycle of length $l \leq d_H(u,v) + d_H(v,a) + d_H(a,u) = 6$ in H. Hence, $N_G[u] \cap N_G[v] \subseteq N_H[w]$ and, consequently, $N_G[u] \cap N_G[v] = N_H[w]$. □

Corollary 4.2. *Let H be a graph of girth at least 7 so that $G = H^2$ is not complete, let v be a vertex with maximum degree in G and let u be a neighbor of v in G but not in H and let w be their common neighbor. Then, for every x in $N_H[w] - \{v\}$, we have that $N_G[v] \cap N_G[x] \neq N_H[w]$ if and only if $x = w$.*

Proof. Every vertex in $N_H(w) - \{v\}$ is a neighbor of v in G but not in H. Thus, by Proposition 4.1 we have that $N_G[v] \cap N_G[x] = N_H[w]$, for any $x \in N_H(w) - \{v\}$. The vertex w is not the only element in $N_H(v)$, otherwise we would have that $N_G[v] = N_H[w]$ and, as v has maximum degree in G, the graph G would be complete. Let a be a vertex in $N_H(v) - \{w\}$. We have that $a \notin N_H[w]$, otherwise H would have a C_3. However, $a \in N_G[w]$, thus $N_G[v] \cap N_G[w] \neq N_H[w]$. □

If a non-complete n-vertex graph has a square root H of girth at least 7, it is possible, based on Corollary 4.2, to determine one edge of H in time $\mathcal{O}(n^2)$. The following algorithm uses this fact, executing algorithm G_6-SqrtEdge at most twice. If G is complete, a solution is a star graph with the same vertices as G, and will be found on the first execution of G_6-SqrtEdge. As the square root with girth at least 6 is unique up to isomorphism, if graph with girth 6 is returned, there is no solution.

```
G_7-Sqrt(G)
Input: a connected graph G with at least 3 vertices
Output: a square root of G with girth at least 7, if it exists;
v ← a maximum degree vertex of G
u ← a neighbor of v
H ← G_6-SqrtEdge(G, uv)
If H = "DOES NOT COMPUTE"
    C ← N_G[v] ∩ N_G[u]
    For each w ∈ C − {v}
        If N_G[v] ∩ N_G[w] ≠ C
            H ← G_6-SqrtEdge(G, vw)
If H ≠ "DOES NOT COMPUTE" and H is C_6-free
    Return H
Return "DOES NOT COMPUTE"
```

Theorem 4.3. *It is possible to decide if an n-vertex graph has a square root of girth at least seven and to compute this root in time $\mathcal{O}(n^2)$.*

Proof. The algorithm G_7-Sqrt(G) solves $\mathcal{G}_7$-square root. In this algorithm, the procedure G_6-SqrtEdge(G, vw), that is $\mathcal{O}(n^2)$, is executed at most twice. Every other step is $\mathcal{O}(n)$ and is executed at most n times. □

Acknowledgements

This work was partially supported by CNPq (Proc. 420079/2021-1).

References

[1] Anna Adamaszek and Michał Adamaszek. Uniqueness of graph square roots of girth six. *The Electronic Journal of Combinatorics [electronic only]*, 18(1):Research Paper P139, 5 p., electronic only–Research Paper P139, 5 p., electronic only, 2011.

[2] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021.

[3] Babak Farzad, Lap Chi Lau, Van Bang Le, and Nguyen Ngoc Tuy. Complexity of finding graph roots with girth conditions. *Algorithmica*, 62(1-2):38–53, February 2012.

[4] Majid Karimi. Square root finding in graphs. Master of science, Faculty of Mathematics and Science, Brock University, St. Catharines, Ontario, 2013.

[5] Rajeev Motwani and Madhu Sudan. Computing roots of graphs is hard. *Discrete Applied Mathematics*, 54(1):81–88, 1994.

[6] Raphael Yuster and Uri Zwick. Finding even cycles even faster. *SIAM Journal on Discrete Mathematics*, 10(2):209–222, 1997.