Computing Roots of Graphs is Hard

Rajeev Motwani *

Department of Computer Science
Stanford University
Stanford, CA 94305-2140

Madhu Sudan †

IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Abstract

The square of an undirected graph G is the graph G^2 on the same vertex set such that there is an edge between two vertices in G^2 if and only if they are at distance at most 2 in G. The k'th power of a graph is defined analogously. It has been conjectured that the problem of computing any square root of a square graph, or even that of deciding whether a graph is a square, is NP-hard. We settle this conjecture in the affirmative.

1. Introduction

We consider the problem of deciding whether a graph is a *perfect square*. Informally, the square of an undirected graph is obtained by placing an edge between two vertices which are at a distance of two or less. It is easy to see that the adjacency matrix of the square graph is the square (under boolean matrix multiplication) of the original adjacency matrix. Our main result is that the problem of deciding whether a graph is a square graph is NP-complete. We start by providing a formal definition of the power of a graph.

Let $G(V, E)$ be an undirected graph on n vertices with m edges. The *distance* between two vertices u and v in G is denoted by $d(u, v)$. The k'th power of the graph G is defined as follows.

Definition 1.1: *For any positive integer k, the graph $G^k(V, E^k)$ has an edge (u, v) if and only if $d(u, v) \leq k$.*

If we assume that G has a self-loop on all the vertices, then it is clear that taking the k'th power of the adjacency matrix of G gives the adjacency matrix of G^k. We will call G a *k'th-root* of the graph G^k. In particular, G^2 is the square of the graph G and G is a square root of the graph G^2.

Powers of graphs have been studied extensively in graph theory. For example, it is well-known that the square of a 2-connected graph has a Hamiltonian cycle [2], and the Hamiltonian cycle

*Supported by Mitsubishi Corporation, NSF Grant CCR-9010517 and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

†Part of this research was done while this author was a student at the University of California at Berkeley, supported by NSF PYI Grant CCR-8896202.

can be found in polynomial time [6]. Furthermore, Sekanina [11] showed that for any non-trivial connected graph G, the graph G^3 is Hamiltonian. Our main motivation for studying the complexity of checking squareness of a graph comes from distributed computing. In this application [8], the t'th power of a graph G represents the possible flow of information during t rounds of communication of a distributed network of processors organized according to G. Our motivation in studying this problem was the question posed by Nati Linial of characterizing the class of graphs that are t'th powers, in the hope that this would facilitate the study of distributed algorithms for graph problems. Our result implies that there does not exist any good (polynomially verifiable) characterization of even the square graphs.

Several attempts have been made at characterizing the class of square graphs. Mukhopadhyay [9] showed that a graph H is a square if and only if there exists a *complete* induced subgraph H_i corresponding to each vertex v_i in H such that

- $v_i \in H_i$.
- $v_i \in H_j$ if and only if $v_j \in H_i$.
- $\cup_i H_i = H$.

This is not a *polynomial* characterization in that it does not lead to a polynomial time algorithm for recognizing square graphs. Similar characterizations were provided for the squares of directed graphs [4] and the k'th powers [1].

Several results have been obtained on the powers of special classes of graphs, or in the case where the power of a graph belongs to a special class. Ross & Harary [10] showed that the tree square roots of a graph, when they exist, are unique up to isomorphisms. Harary, Karp & Tutte [5] provided characterizations of the planar graphs which are square graphs. Recently, Lin & Skiena [7] devised several algorithms with respect to powers of graphs. They provided efficient algorithms for finding square-roots of graphs G^2 where G is a tree, and where G^2 is planar. They also presented several polynomial-time algorithms for problems which are NP-complete in general, when restricted to the special case of graphs that are k'th powers. Lin & Skiena conjectured that the problem of recognizing square graphs is NP-complete – we settle this conjecture in this paper. We believe that the square-testing problem remains NP-complete even when restricted to the special case of squares of bipartite graphs, but our proof does not seem to extend to this problem. An important related problem that remains open is the issue of finding square roots of matrices under field operations. These questions bear upon the construction of block designs and related combinatorial structures.

2. Preliminaries

Throughout this paper ϕ will represent a 3-CNF formula on n variables $x_1, \cdots, x_n$ with m clauses. The n variables give $2n$ literals and we will use $l_1, \cdots, l_{2n}$ to denote them.

Definition 2.1: *A 3-CNF formula ϕ has a* **not-all-equal** *satisfying assignment, if there exists an assignment to the variables $x_1, \cdots, x_n$ such that each clause of ϕ has at least one* `TRUE` *literal and at least one* `FALSE` *literal.*

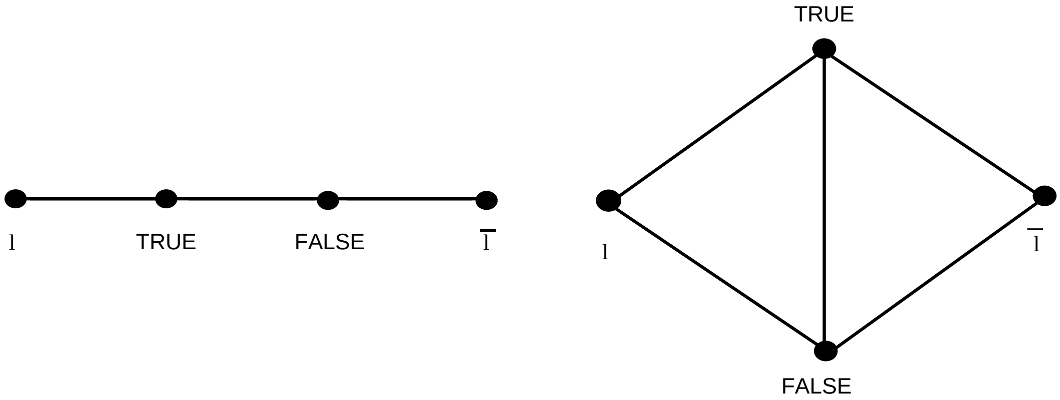

Figure 1: P_4 and P_4^2

Definition 2.2: NAESAT *is the set of all 3-CNF formulae that have not-all-equal satisfying assignments.*

The following result can be found in the book of Garey & Johnson [3].

Theorem 1: *Given a 3-CNF formula ϕ, determining if $\phi \in$* **NAESAT** *is NP-Complete.*

Problem [SQUARE]
Instance: *A graph $G = (V, E)$.*
Question: *Does there exists a graph H such that $G = H^2$?*

The rest of the paper shows that **SQUARE** is NP-hard by reducing **NAESAT** to it. It is clear that **SQUARE** is in NP, since guessing the square root H and verifying that $G = H^2$ can be easily done in polynomial time. Thus, we conclude that **SQUARE** is NP-complete.

Theorem 2: SQUARE *is NP-Complete*

3. Intuitive description of the reduction

This section describes the basic ideas and tools that set up the reduction from **NAESAT** to **SQUARE**. The reduction starts off by trying to set up a graph H which represents a satisfying assignment to a 3-CNF formula such that its square G would effectively hide all information about the assignment. Furthermore, the reduction would try to maintain that any square root of G gives a satisfying assignment to the formula.

One of the most important insights into the problem is obtained by looking at a very simple graph – the path on 4 vertices (P_4, see Figure 1). The square of this graph is the complete graph on 4 vertices from which one edge is missing. Associate with the endpoints of this path the literals l and $\bar{l}$, and associate with the two inner vertices the constants `TRUE` and `FALSE`. This association can be used to represent an assignment to the variable l : l is `TRUE` if and only if l is adjacent to `TRUE`. In the square of this graph the literals l and $\bar{l}$ are both connected to the constants `TRUE` and `FALSE`, so the square no longer contains any information of the assignment to l. At the same time,

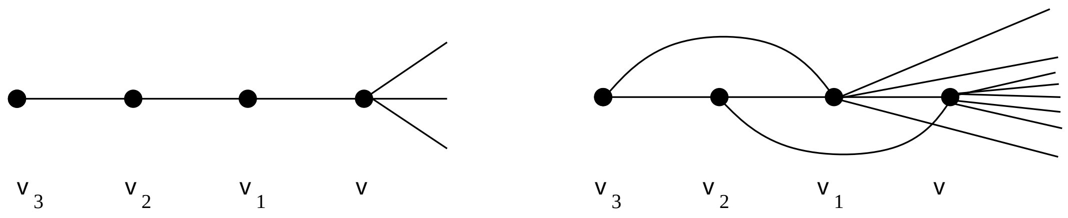

Figure 2: Tail in H and H^2

enough information is preserved to ensure that any square root of this graph would have been some assignment to l.

The natural direction for the reduction would be to replicate this process with many literals. However the notion of TRUE and FALSE has to be global – so the reduction would have to use exactly one pair of TRUE and FALSE vertices and all literals that are assigned TRUE will be made adjacent to TRUE in H (similarly for FALSE literals). There is a problem with this process - all TRUE literals become adjacent in the square. This problem though is overcome easily by making all literal pairs l_1 and l_2 adjacent in G. This is achieved by creating for every pair of literals l_1, l_2, which are not complements of each other, a vertex $A_{l_1 l_2}$ which is adjacent to l_1 and l_2 in H. By throwing in a few more edges connecting $A_{l_1 l_2}$ to some other vertices, it can be ensured that the square of H has no information at all about the assignment.

The next step would be to force the assignment to be a not-all-equal satisfying assignment for a set of clauses $\mathcal{C}$. This is achieved by creating for each clause a vertex c which is adjacent in H to all literals contained in the clause. The fact that the clause contains at least one and at most two TRUE literals, implies that in G, c will be connected to both TRUE and FALSE.

Lastly we need a gadget which can be used to enforce some nice properties of any square root of the square of a fixed graph H. One such property might be to ensure that some vertex v has the same neighborhood in any square root of H^2 as in H. This is enforced by adding to v, in H, a tail, which is a sequence of vertices $v_3 \leftrightarrow v_2 \leftrightarrow v_1$ where v_1 is adjacent only to v (see Figure 2). It can be argued by looking at H^2 that in any square root of H^2, v_3 is adjacent only to v_2, which in turn is adjacent only to v_1 and so on, finally enabling one to exactly pin down the neighborhood of v.

The next section gives the complete and exact details of the reduction, along with a formal proof of its correctness.

4. The Reduction

Let c_j be the set of literals in clause j and let $\mathcal{C} = \{c_j | 1 \leq j \leq m\}$. The graph G is constructed as follows :

Vertices of G

- Constant Vertices : TRUE, FALSE, $\mathcal{X}$ and tail vertices to $\mathcal{X} - t_1$, t_2 and t_3.
- Literal Vertices : $L_i : 1 \leq i \leq 2n$ for each literal l_i.

- Literal Pair Vertices : $A_{ij} : 1 \leq i < j \leq 2n$ and $l_i \neq \bar{l}_j$.
- Clause Vertices : C_j for each clause $c_j \in \mathcal{C}$ and tail vertices C_j^1, C_j^2, C_j^3 for each c_j.

Edges of G

- Edges of the tail of $\mathcal{X}$: $t_3 \leftrightarrow t_2$, $t_3 \leftrightarrow t_1$, $t_2 \leftrightarrow t_1$, $t_2 \leftrightarrow \mathcal{X}$, $t_1 \leftrightarrow \mathcal{X}$, $t_1 \leftrightarrow$ `TRUE`, and $t_1 \leftrightarrow$ `FALSE`.
- Edges of the tails of clauses : $\forall c_j \in \mathcal{C}$, $C_j^3 \leftrightarrow C_j^2$, $C_j^3 \leftrightarrow C_j^1$, $C_j^2 \leftrightarrow C_j^1$, $C_j^2 \leftrightarrow C_j$, $C_j^1 \leftrightarrow C_j$, $C_j^1 \leftrightarrow L_{i_1}$, $C_j^1 \leftrightarrow L_{i_2}$, and $C_j^1 \leftrightarrow L_{i_3}$, where $\{l_{i_1}, l_{i_2}, l_{i_3}\} = c_j$.
- Edges of the clauses : $\forall c_j \in \mathcal{C}$, and $\forall i$ s.t. $l_i \in c_j$, $C_j \leftrightarrow L_i$; $C_j \leftrightarrow$ `TRUE`, $C_j \leftrightarrow$ `FALSE`. Also for all k s.t. $l_k \neq \bar{l}_i$, $C_j \leftrightarrow A_{ik}$.
- Edges of the literal vertices, literal pair vertices and the vertices `TRUE`, `FALSE`, $\mathcal{X}$: All remaining vertices $\{L_i\}$'s, the $\{A_{ij}\}$'s and `TRUE`, `FALSE`, $\mathcal{X}$ are adjacent, *except* for the pairs, L_i, L_j where $l_i = \bar{l}_j$.

The important thing about G_ϕ is that it can be constructed without any knowledge of the satisfiability of ϕ. At the same time G_ϕ contains all the information of ϕ. The next two subsections show that G_ϕ is a square if and only if ϕ is satisfiable in a not all equal manner.

4.1. Satisfiability implies squareness

Here we prove that if $\phi \in$ **NAESAT**, then G_ϕ is a square. Consider an assignment to the variables $x_1, \cdots, x_n$, such that every clause of ϕ contains at least one true and one false literal. A square root of G_ϕ, say H, can be constructed as follows :

Edges of H

- Edges of $\mathcal{X}$ and the tail of $\mathcal{X}$: $t_3 \leftrightarrow t_2$, $t_2 \leftrightarrow t_1$, $t_1 \leftrightarrow \mathcal{X}$ and $\mathcal{X} \leftrightarrow$ `TRUE`, `FALSE`.
- Edges of clause vertex C_j and its tail : $C_j^3 \leftrightarrow C_j^2$, $C_j^2 \leftrightarrow C_j^1$, $C_j^1 \leftrightarrow C_j$ and $C_j \leftrightarrow L_i$ if and only in $l_i \in c_j$.
- Edges of literal vertex L_i : Here we use the assignment information and connect L_i to `TRUE` if it is true under the assignment and connect it to `FALSE` otherwise. Also for each literal k s.t. $l_k \neq \bar{l}_i$, $L_i \leftrightarrow A_{ik}$.
- Edges of literal pair vertex A_{ik} : $A_{ik} \leftrightarrow$ `TRUE`, `FALSE`.

Lemma 3: $G_\phi = H^2$

Proof: It can be verified that the adjacency of the tail vertices of $\mathcal{X}$ and the tail vertices of the clause vertices have the same adjacency in G_ϕ as in H^2. Also, the clause vertex C_j has as its neighbors in H^2 all neighbors in H of literal vertices L_{i_1}, L_{i_2} and L_{i_3} where $c_j = \{l_{i_1}, l_{i_2}, l_{i_3}\}$. This can be used to check that the adjacency of the clause vertices is the same in H^2 as in G_ϕ.

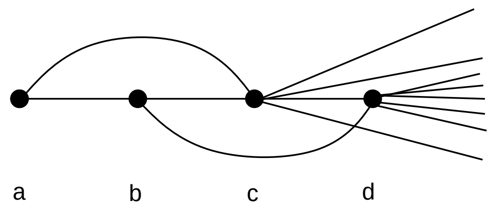

Figure 3:

Consider the induced subgraph on the literal vertices, the literal pair vertices and the constant vertices. Since no literal and its complement share a common neighbor in H, vertex pairs corresponding to a literal and its complement are non-adjacent in H^2. All other vertices get connected mainly because they either share a common neighbor in `TRUE` or `FALSE`, or because of the existence of the literal pair vertices.

We have omitted some details for the sake of clarity, but these can be easily verified. □

4.2. Squareness implies satisfiability

We now show that if G is a square, then $\phi \in$ **NAESAT**. First we derive a property of the "tails" which are crucial to our construction.

Lemma 4: *If a, b, c, d are vertices of G such that*

- *The only neighbors of a are b and c.*
- *The only neighbors of b are a, c and d.*
- $c \leftrightarrow d$

Then the neighbors, from $V - \{a, b, c, d\}$, of d in any square root of G are the same as the neighbors, from $V - \{a, b, c, d\}$, of c in G.

Proof: Assume, without loss of generality, that G is a connected graph and let H be a square root of G. Then H is connected and hence G is biconnected. Hence both c and d must have neighbors in $V - \{a, b, c, d\}$ in G (see Figure 3).

First, observe that the degree of a in H must be one, or else d would have to be a neighbor of a in G. If $c \leftrightarrow a$ in H then c can only be adjacent to b and a in H. But c has neighbors in $V - \{a, b, c, d\}$ in G while a and b do not. This implies that the only neighbor of a in H is b. Now the following line of reasoning pins down the neighborhood of a, b, c, d exactly :

- b is adjacent only to c (besides a) in H : This follows by looking at the neighborhood of a in G.
- c is adjacent only to d (besides b) in H : This follows by looking at the neighborhood of b in G.

- d is adjacent only to the neighbors of c in $V - \{a, b, c, d\}$ (besides c) in H : Once again this follows by looking at the neighborhood of c in G.

$\square$

Immediate corollaries to this lemma are the following :

Corollary 5: *If H_ϕ is a square root of G_ϕ then $\mathcal{X}$ is adjacent only to* `TRUE` *and* `FALSE` *in H_ϕ (besides neighbors in its tail).*

Corollary 6: *If H_ϕ is a square root of G_ϕ then C_j is adjacent only to L_{i_1}, L_{i_2} and L_{i_3} in H_ϕ where $c_j = \{l_{i_1}, l_{i_2}, l_{i_3}\}$ (besides neighbors in its own tail).*

Corollary 5 forces H_ϕ to be an assignment. Since every literal vertex is adjacent to $\mathcal{X}$ in G_ϕ, every literal must be adjacent to at least one of `TRUE` or `FALSE` in H_ϕ. But no literal vertex is adjacent to its complementary vertex in G_ϕ. Thus every literal vertex must be adjacent to exactly one of `TRUE` and `FALSE` and its complement must be adjacent to the other vertex. This forces H_ϕ to look like an assignment.

Now Corollary 6 can be used to force this to be a not-all-equal satisfying assignment. Every clause vertex sees both `TRUE` and `FALSE` in G_ϕ but it sees only its own literals in H_ϕ. Therefore at least one of the literals that a clause sees must be true and at least one must be false. This implies that the assignment given by H_ϕ is a not-all-equal satisfying assignment.

This completes the proof that G_ϕ is a square if and only if $\phi \in$ **NAESAT**.

5. Acknowledgments

We would like to thank Nati Linial for posing the problem over lunch at the Jordan Hall Cafe.

References

[1] F. Escalante, L. Montejano and T. Rojano, Characterization of n-path graphs and of graphs having n'th root, *Journal of Combinatorial Theory (Series B)*, 16 (1974), pp. 282–289.

[2] H. Fleischner, The square of every two-connected graph is Hamiltonian, *Journal of Combinatorial Theory (Series B)*, 16 (1974), pp. 29–34.

[3] M.R. Garey and D.S. Johnson, "Computers and Intractability – A Guide to the Theory of NP-completeness," W.H. Freeman & Co., 1979.

[4] D.P. Geller, The square root of a digraph, *Journal of Combinatorial Theory (Series B)*, 5 (1968), pp. 320-321.

[5] F. Harary, R.M. Karp and W.T. Tutte, A criterion for planarity of the square of a graph, *Journal of Combinatorial Theory (Series B)*, 2 (1967), pp. 395–405.

[6] H.T. Lau, "Finding a Hamiltonian Cycle in the Square of a Block," PhD. Thesis, School of Computer Science, McGill University, Montreal, 1980.

[7] Y-L. Lin and S.S. Skiena, "Algorithms for Square Roots of Graphs," Technical Report TR# 91/11, Department of Computer Science, SUNY Stony Brook, 1991.

[8] N. Linial, Locality in distributed graph algorithms, *SIAM Journal on Computing*, 21 (1992), pp. 193–201.

[9] A. Mukhopadhyay, The square root of a graph, *Journal of Combinatorial Theory (Series B)*, 2 (1967), pp. 290-295.

[10] I.C. Ross and F. Harary, The square of a tree, *Bell System Technical Journal* 39 (1960), pp. 641–647.

[11] M. Sekanina, "On an ordering of the set of vertices of a connected graph," Technical Report No. 412, Publ. Fac. Sci. Univ. Brno, 1960.