The Complexity of Boolean Matrix Root Computation

Martin Kutz[*]

Freie Universität Berlin, Germany
kutz@math.fu-berlin.de

Abstract. We show that finding roots of Boolean matrices is an $\mathcal{NP}$-hard problem. This answers a twenty year old question from semigroup theory. Interpreting Boolean matrices as directed graphs, we further reveal a connection between Boolean matrix roots and graph isomorphism, which leads to a proof that for a certain subclass of Boolean matrices related to subdivision digraphs, root finding is of the same complexity as the graph-isomorphism problem.

1 Introduction

Multiplication of Boolean zero-one matrices is defined as ordinary matrix multiplication with $+$ and $\cdot$ replaced by the Boolean operations $\vee$ and $\wedge$. So the matrix product $C = AB$ is given by

$$c_{ij} = \bigvee\nolimits_{h=1}^{n} a_{ih} \wedge b_{hj},$$

and as with matrices over fields, the kth power A^k of a Boolean $n \times n$ matrix A is simply the k-fold product of A with itself.

Besides its theoretical relevance for semigroup theory, Boolean matrix algebra serves as a fundamental tool in algorithmic graph theory. Efficient algorithms for transitive-closure or shortest-path computations rely on the interpretation of directed graphs as Boolean matrices [16,1,3].

In this work, we investigate the computational complexity of finding roots of a given Boolean matrix. A kth *root* of a square Boolean matrix B is some other matrix A whose kth power A^k equals B. Twenty years ago, in the open problems section of his book [9], Kim asked if given a matrix B, such a root A can be computed in polynomial time or whether this problem is perhaps $\mathcal{NP}$-complete. (Actually, he inquired for the case $k = 2$ only.) We give an answer to that question.

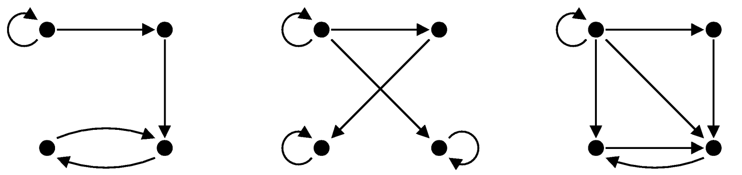
Theorem 1. *Deciding whether a square Boolean matrix has a* kth *root is* $\mathcal{NP}$*-complete for each single parameter* $k \geq 2$.

[*] Member of the European graduate school "Combinatorics, Geometry, and Computation" supported by the Deutsche Forschungsgemeinschaft, grant GRK 588/2.

With the “right” computational problem for the reduction, the proof of this result turns out surprisingly simple. This is quite remarkable since it thus relates Boolean matrix roots to a well-known $\mathcal{NP}$-complete problem, which yields insight in the local structure of Boolean matrices.

In the second, technically more challenging part of our work, we reveal further properties of matrix roots which show a close relation to graph isomorphism. This eventually leads to a proof that for a certain subclass of Boolean matrices kth root computation is graph-isomorphism complete. Before we can state this result precisely, we have to switch from matrices to the graph theoretic point of view. Actually, throughout this whole exposition we shall interpret Boolean matrices as adjacency matrices of directed graphs.

Boolean Matrices and Graph Theory. Any Boolean $n \times n$ matrix $A = (a_{ij})$ can be interpreted as a directed graph D on the vertex set $\{1, \ldots, n\}$ with an arc from j to i iff $a_{ij} = 1$. So D may have loops but no multiple arcs. The kth power of D, $k \in \mathbb{N}$, is the directed graph D^k defined on the same vertex set and with an arc from a to b if and only if there is a directed walk of length exactly k from a to b in D (possibly visiting some vertices several times); compare the figure. It is easy to see that the adjacency matrix of D^k is in fact the kth power of the adjacency matrix of D (see, for example, [18]).[1]



So taking the graph theoretic point of view, we investigate the kth-root problem for digraphs: *given a directed graph D, does there exist another digraph R (on the same vertex set) such that $R^k = D$.* Our answer to the guiding question then reads as follows.

Theorem 1 (digraph version). *Deciding whether a digraph has a* kth *root is $\mathcal{NP}$-complete for each single parameter $k \geq 2$.*

Our second main result, which relates roots to isomorphisms, is based on subdivisions, defined as follows.

Definition 1. *The* complete subdivision *of a digraph D is the digraph obtained from D by replacing each arc $a \to b$ of D by a new vertex x_{ab} and the two arcs $a \to x_{ab} \to b$. We call a digraph a* subdivision digraph *if it is (isomorphic to) the complete subdivision of some digraph.*

[1] Alternatively, one might view a Boolean matrix as a binary relation. Then the kth matrix power is simply the k-fold composition of this relation.

Subdivisions are a fundamental notion in graph theory. But opposed to their common usage in relation with topological minors, we employ them here to equip our graphs with a certain stiffness that makes root finding computationally simpler. In fact, under an additional minor degree condition on which we shall comment later, we can show that finding roots of such graphs is of the same complexity as the graph-isomorphism problem.

Theorem 2. *Deciding whether a subdivision digraph with positive minimal indegree and outdegree has a kth root, is graph-isomorphism complete for each parameter $k \geq 2$.*

Graph Isomorphism. The graph-isomorphism problem asks: *are two given (di)-graphs*[2] *isomorphic or not?* It is neither known to have a polynomial-time solution nor is it known to be $\mathcal{NP}$-complete. On the contrary, it is a prime candidate for a problem strictly between $\mathcal{P}$ and $\mathcal{NP}$-completeness (cf. [10] and [12]). A computational problem of the same complexity as the graph-isomorphism problem is called *graph-isomorphism complete*, or simply *isomorphism complete* because isomorphism problems for several algebraic or combinatorial structures fall into this class. For example, isomorphism of semigroups and finite automata [2], finitely represented algebras, or convex polytopes [8]. Other problems ask for properties of the automorphism group of a graph, for example, computing the size of the automorphism group or its orbits [14].[3] Finally, several restrictions of the graph-isomorphism problem are known to remain isomorphism complete, as for example isomorphism of regular graphs [2].

As the above list indicates, actually all problems known to be isomorphism complete are more or less obviously isomorphism problems of various combinatorial structures. Hence, the relation between digraph roots and graphs isomorphism established through Theorem 2 may come quite as surprise.

Theorem 2 rests on a structural result which states that any kth root of a subdivision digraph D essentially establishes a one-to-one correspondence between k isomorphic subgraphs of D (Theorem 3). Due to space constraints, we shall only sketch the proofs of Theorems 1, 2, and 3, stating some of the central lemmas that pave the way.

2 Related Work—Related Questions

Over the field of complex numbers or the reals, matrix roots are a well-studied and still up-to-date topic of linear algebra [11,7,17]. But results from that field of research do generally not apply to Boolean matrices. While it is known, for example, that every regular matrix over the complex numbers has a kth root for

[2] One usually considers undirected graphs but it is well-known and easily seen that with respect to their computational complexity the undirected and directed version of the problem are equivalent.

[3] The latter two problems are known to be isomorphism complete only in the weaker sense of Turing reduction, as opposed to the concept of many-one reduction.

any $k \geq 2$ [17], this is not true for Boolean matrices, as the invertible matrix $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ shows. Further, complex or real matrices are amenable to numerical methods like Newton iteration [6], whereas such techniques clearly do not apply to Boolean matrices. When it comes to roots, Boolean matrices don't seem to have much in common with matrices over $\mathbb{C}$ since the former behave much more rigidly than the latter.

The situation is, however, slightly different if we ask for powers of a matrix instead of roots. There are theoretical results on Boolean matrix powers [4] and in practice we can of course compute the kth power of a Boolean matrix A by treating it as a matrix over the reals. We calculate A^k over $\mathbb{R}$ and afterwards replace each positive entry with 1. This simple reformulation allows us, for example, to apply fast matrix multiplication methods such as Strassen's to path problems in graphs [16,1]. But this simulation through matrices over the reals clearly only works because there cannot happen cancellation between positive and negative entries. For root finding, such simulation over $\mathbb{R}$ or $\mathbb{C}$ would lead into major problems.

Alternative Notions of Graph Powers. A problem similar to the one at hand has been discussed by Motwani and Sudan. In [15] they showed that computing square roots of undirected graphs is $\mathcal{NP}$-hard. But their notion of graph powers differs from ours in two important points.

They consider undirected graphs only, which corresponds to having bidirectional edges in our setting. This not only restricts the set of possible inputs but also—and this is the decisive difference—the solutions. For example, the four-vertex graph consisting of two disjoint bidirectional arcs has the directed 4-cycle as a square root, but no undirected graph can be a root.

Further, Motwani and Sudan define squaring to maintain existing edges, which in our setting would corresponds to attaching loops to all vertices. This monotonicity ensures that much information of the underlying graph can be read off from its square and the hardness proof of [15] makes essential use of this property. In contrast to this, squaring a digraph under the rules derived from Boolean matrix multiplication can almost completely destroy the neighborhood information and may even decompose the digraph. Actually, most of our arguments depend crucially on such "vanishing edges." So apparently, the squares in [15] and our notion of powers are essentially different concepts.

3 $\mathcal{NP}$-Completeness

We now sketch the proof of Theorem 1, presenting the two main ingredients for our $\mathcal{NP}$-completeness result: a suitable $\mathcal{NP}$-complete problem and a many-one reduction from that problem to digraph roots.

Surprisingly, the reduction is very straightforward, it goes without any sophisticated gadgets. This simplicity indicates that the two problems are actually very closely related. While skipping the details of the correctness proof here, we elaborate a bit on the reduction to visualize and endorse this claim. The appropriate problem for our reduction is the *set-basis problem:*

Let $\mathfrak{C}$ be a collection of subsets of some finite set S. A *set basis* of $\mathfrak{C}$ is another collection $\mathfrak{B}$ of subsets of S such that each $C \in \mathfrak{C}$ can be written as a union of sets from $\mathfrak{B}$. Given a finite set S together with such a collection $\mathfrak{C}$ of subsets of S and an integer $r \leq |S|$, the *set-basis problem* asks whether there exists a set basis $\mathfrak{B}$ for $\mathfrak{C}$ consisting of at most r sets. This problem is known to be $\mathcal{NP}$-complete [19].

The Reduction. The key idea for the reduction stems from the following general observations about digraph square roots.

Consider some set X of vertices of a digraph D and let Z denote all outneighbors of vertices in X. Let us assume for simplicity that X and Z are disjoint, so in particular, there are no loops or cycles. Then in a square root of the digraph D, any of the arcs from X to Z must be realized as walks of length two and since there are no loops, these walks are actually paths. Hence, in the root there must exist a set Y of "intermediate vertices" through which all these paths can pass. If now—for whatever reason—there is only a small number of such intermediate vertices available, $|Y| \leq r$, say, with r a little smaller than $|Z|$, these paths have to interact in order to ship all their information from X to Z.

We claim that the the square-root problem for these sets X, Y, and Z is nothing but a set-basis instance. This is easily seen by interpreting Z as the ground set S and X as a collection of subsets of S, defined trough containment relations given by the original D-arcs. The vertices in Y represent the set basis where the root arcs from Y to Z define the subsets and the arcs from X to Y tell us how to represent sets in X as unions of Y-sets.

In order to turn a set-basis instance into a digraph, we simply draw the containment graph of the set system $\mathfrak{C}$ on S and provide the right number of intermediate vertices. In the general case of kth roots that would be $k-1$ times r vertices, which we leave almost isolated except for some framework arcs to ensure that any root uses them as intended.

Interpretation. We emphasize that the given set-basis instance is completely maintained by our reduction. Its containment relations are encoded one-to-one by arcs of the digraph. Moreover, the preceding discussion shows that an instance of the digraph-root problem can be seen as a large collection of interacting set-basis problems. One might well argue that finding digraph roots is actually a generalized set-basis problem.

As a corroboration for this point of view we mention that the set-basis problem already appeared before in connection with Boolean matrix algebra. Markowsky [13] used it in a very economic proof for the $\mathcal{NP}$-completeness of Schein-rank computation.[4]

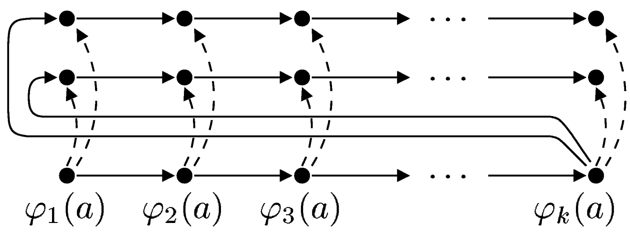
[4] Analogous to the matrix rank over fields, the *Schein rank* of a Boolean matrix A is the minimal integer ρ such that A can be represented as a Boolean sum $A = \bigvee_{i=1}^{\rho} c_i r_i$, where the c_i are column and the r_i row vectors with zero-one entries [9, Sec. 1.4].

4 Roots and Isomorphism

In this second part, we establish the isomorphism-completeness result of Theorem 2. Our considerations are guided by the following fundamental connection between digraph roots and digraph isomorphism.

Proposition 1. *Let $D = D_1 \dot{\cup} D_2 \dot{\cup} \cdots \dot{\cup} D_k$ be the disjoint union of k isomorphic digraphs $D_1, \ldots, D_k$. Then D has a k*th *root.*

Because the proof is short, instructive, and of importance for the general understanding of Theorem 2, we briefly sketch the ideas. We construct the sought-after root R on the vertices of D from the isomorphisms $\varphi_i\colon D_1 \to D_i$, $1 \leq i \leq k$ (φ_1 being simply the identity). For each vertex a of D_1, we let R contain the path $\varphi_1(a) \to \varphi_2(a) \to \cdots \to \varphi_k(a)$ and additionally the arcs $\varphi_k(a) \to b$ for all D-outneighbors b of a. The following figure shows a local picture of this construction. (The continuous lines form the root, the dashed lines the given D.) One easily verifies that in fact, $R^k = D$.



Obviously it was essential to switch from matrices to digraphs. While it might be possible to carry out our $\mathcal{NP}$-completeness proof in terms of matrices, the statement and proof of Proposition 1 clearly belong to the realm of graph theory.

Identifying Subdivision Vertices

The crucial step towards our isomorphism-completeness result is to show that subdivision digraphs almost satisfy a converse of Proposition 1. That is, any root of such a digraph carries an isomorphism structure of its components. However, we have to take care of some degenerate cases that do not fit into this picture.

Usually in a subdivision digraph, one can easily distinguish the original vertices, commonly called the *branching vertices*, from the *subdivision vertices.* In fact, a subdivision digraph is obviously bipartite and as soon as every weakly connected component contains at least one vertex whose indegree or outdegree differs from 1, the two classes can be uniquely identified.

A problem arises with subdivision digraphs that contain isolated cycles (of even length). In such components, all vertices look like subdivision vertices and this absence of clearly identifiable branching vertices leads to untypical behavior with respect to root finding. Fortunately, isolated cycles are simple objects and we can completely describe their powers.

Lemma 1. *The* kth *power of a directed cycle of length* r *is the disjoint union of* $\gcd(r, k)$ *cycles of length* $r/\gcd(r, k)$*.*

As a consequence, isolated cycles clearly cannot have the isomorphism property we are looking for. But this is no problem. It turns out that a vertex that lies on an isolated cycle of a subdivision digraph D must also lie on an isolated cycle in any root of D. Thus, with respect to roots, cycle vertices do not interact with the other vertices of a subdivision digraph and so we may in the following restrict our attention to subdivision digraphs without cycles. Then each vertex can really be uniquely identified as subdivision or branching vertex.

From Roots to Isomorphisms

With all isolated cycles removed, subdivision digraphs now bear the desired isomorphism structure, under the unfortunately indispensable additional condition that each vertex has at least one inneighbor and one outneighbor.

Theorem 3. *A subdivision digraph without isolated cycles and with positive minimal indegree and outdegree has a* kth *root if and only if it is the disjoint union of* k *isomorphic digraphs.*

The proof of Theorem 3 is lengthy and rather technical and we have to omit the details due to space constraints, but the key ideas are easily explained: Recall the long root paths in the proof of Proposition 1. Each inner vertex had indegree and outdegree exactly 1 in R. Theorem 3 rests on the fact that conversely, any root of a subdivision digraph satisfying the preconditions has exactly this structure, i.e., it consists mainly of paths of length $k-1$ on which no other paths enter or leave. From those paths one can read off the desired isomorphisms. Let us substantiate these ideas by stating some of the central lemmas.

Long Paths. Our aim is to assign each vertex of R to a path of exactly k vertices, the beginning and end of which shall be uniquely determined.

Lemma 2. *Let* R *be a* kth *root of a subdivision digraph* D *without isolated cycles. Then any subdivision vertex of* D *lies on an* R*-path* $a_1 \to a_2 \to \cdots \to a_k$ *of length* $k-1$ *where each* a_i*,* $1 \le i \le k$*, is a subdivision vertex of* D*. Moreover, such a path is maximal in the sense that the inneighbors of* a_1 *and the outneighbors of* a_k *are branching vertices of* D*.*

There exists an analog of Lemma 2 for branching vertices, which looks almost the same, with the exception that we have to forbid isolated vertices. So here the degree condition of Theorem 3 enters the first time, in a weakened form.

An attempt to prove Lemma 2 directly, faces a principal problem: *subdivision vertex* and *branching vertex* are global notions. A branching vertex with indegree and outdegree 1 is locally indistinguishable from a subdivision vertex. We resolve this ambiguity by ignoring the global picture for a moment, calling a vertex *thin* if it looks like a subdivision vertex, i.e., if it has indegree and outdegree 1 in D. For such vertices, we can prove a preliminary version of Lemma 2.

Lemma 3. *Let R be a kth root of a subdivision digraph D and let $a_0 \to a_1 \to \cdots \to a_l$ be an R-walk of length $l \leq k$ between two D-thin vertices a_0 and a_l. Then all a_i, $1 \leq i < l$, are also thin (with respect to D).*

The arguments employed in the proof of Lemma 3 are typical for most of our intermediate results on the way to Theorem 3. Thinness in D tells us that all R-walks starting from a_0 and a_l have to meet again after exactly k steps. We use these confluent walks to "sandwich" R-walks that start from one of the intermediate vertices a_i, showing that those walks also meet again after a certain number of steps. Thinness of a_i in D finally follows from the simple but important observation that in a subdivision digraph, two different vertices cannot have common inneighbors and common outneighbors at the same time.

Combining Lemma 3 with its analog for non-thin vertices eventually leads to a proof of Lemma 2 and its counterpart for branching vertices.

Unique Arcs. In order to use the paths from the preceding paragraph for isomorphism construction, we have to make sure that they indeed establish one-to-one correspondences. Therefore we show that those paths do not interfere, i.e., they must only touch at their end vertices. As above, we resort to the technical notion of thinness.

Lemma 4. *Let R be a kth root of a subdivision digraph D and let a, b be two thin vertices of D with $a \to b$ in R. Then there are no further R-arcs leaving a or entering b.*

Again, we have an analog of this lemma for pairs of non-thin vertices but once more we have to be careful about the existence of neighbors, which was trivially guaranteed for thin vertices. In the next lemma, the additional degree condition of Theorem 3 is indispensable.

Lemma 5. *Let R be a kth root of a subdivision digraph D. Let a, b be two non-thin vertices of D with $a \to b$ in R such that a has an outneighbor and b has an inneighbor in D. Then there are no further R-arcs leaving a or entering b.*

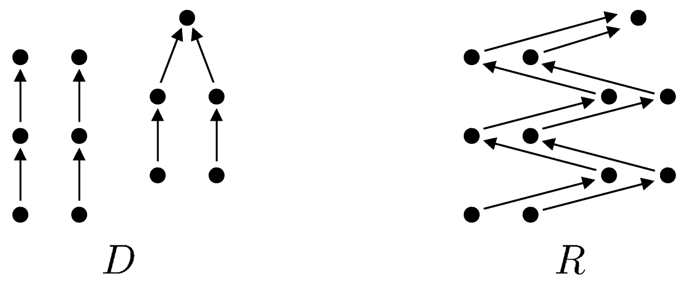
The two preceding lemmas naturally lift to statements about subdivision and branching vertices, the only problem being to show that of two adjacent branching vertices either both are thin or neither is, which is not too difficult.

Concluding the Proofs. The statement of Theorem 3 is now obtained by "inverting" the proof of Proposition 1. The paths provided by Lemma 2 establish correspondences between k disjoint subgraphs of the digraph D and with the help of Lemmas 4 and 5 this can be done in a unique and consistent way.

Theorem 2 then comes as a direct consequence. Isolated cycles are computationally easy to deal with, as we already argued. We should specify that the reduction from root finding to isomorphism finding is actually a Turing reduction, which means that we can find roots in polynomial time on a Turing machine that may call an isomorphism oracle several times at unit cost. Note that we cannot turn it into a stronger many-one reduction (Karp reduction) by simply checking whether k copies of one component of the given digraph D are isomorphic to D itself because the k isomorphic subgraphs of D need not be connected. The other reduction, from isomorphisms to roots, however, can be done in a many-one fashion as Proposition 1 shows.

Dropping the Degree Condition

Let us indicate what can happen in a subdivision digraph that contains vertices without in– or outneighbors. The following figure shows such a digraph D together with a square root R.



The two topmost root arcs can touch since the precondition of Lemma 5 does not hold. Observe that instead of being the disjoint union of two isomorphic subgraphs, the left component can be decomposed into two parts, A and B (the former consisting of the two paths on the left, the latter containing the remaining five vertices) such that there exists a surjective *homomorphism* from A onto B (i.e., an arc-preserving map). This homomorphism corresponds exactly to those arcs of R that go from A to B.

This example is only meant to indicate the phenomena that might show up. The general situation is more difficult to analyze and it is not clear whether the digraph root problem remains isomorphism complete under these relaxed conditions since the general homomorphism problem for graphs is $\mathcal{NP}$-complete [5].

5 Outlook

While the original problem, the open complexity status of Boolean matrix root computation is now settled, the discovered relation to graph isomorphism raises new questions. First of all, it would be desirable to get rid of the degree condition of Theorem 2. Though this restriction turned out indispensable for underlying structural statement of Theorem 3, it is not clear whether it might be possible to eliminate it from the complexity result since that would lead to special homomorphism problems which take us closer to the world of $\mathcal{NP}$-hardness.

More generally, we may ask for relaxations of the concept of subdivisions that still serve the task of "deactivating" computationally hard aspects of root finding, thus keeping the problem isomorphism complete. If one traces the details of our proofs, notions like bounded tree width appear promising and might lead to weaker conditions for isomorphism completeness. Eventually, the problem of Boolean matrix root computation could turn out to be a suitable object for analyzing the "boundary" between isomorphism completeness and $\mathcal{NP}$-hardness.

References

1. A. V. Aho, M. R. Garey, and J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.
2. Kellogg S. Booth. Isomorphism testing for graphs, semigroups, and finite automata are polynomially equivalent problems. *SIAM Journal on Computing*, 7(3):273–279, 1978.
3. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*, chapter 26. MIT Press, 1990.
4. Bart de Schutter and Bart de Moor. On the sequence of consecutive powers of a matrix in a Boolean algebra. *SIAM Journal on Matrix Analysis and Applications*, 21(1):328–354, 1999.
5. Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.
6. Nicholas J. Higham. Newton's method for the matrix square root. *Mathematics of Computation*, 46(174):537–549, 1986.
7. C. R. Johnson, K. Okubo, and R. Reams. Uniqueness of matrix square roots and an application. *Linear Algebra and Applications*, 323:52–60, 2001.
8. Volker Kaibel and Alexander Schwartz. On the complexity of isomorphism problems related to polytopes. To appear in *Graphs and Combinatorics*.
9. Ki Hang Kim. *Boolean matrix theory and applications*. Marcel Dekker, Inc., 1982.
10. Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem*. Birkhäuser, 1993.
11. Ya Yan Lu. A Padé approximation method for square roots of symmetric positive definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 19(3):833–845, 1998.
12. Anna Lubiw. Some NP-complete problems similar to graph isomorphism. *SIAM Journal on Computing*, 1981.
13. George Markowsky. Ordering D-classes and computing Shein rank is hard. *Semigroup Forum*, 44:373–375, 1992.
14. Rudolph Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8(3):131–132, 1979.
15. Rajeev Motwani and Madhu Sudan. Computing roots of graphs is hard. *Discrete Applied Mathematics*, 54(1):81–88, 1994.
16. Ian Munro. Efficient determination of the transitive closure of a directed graph. *Information Processing Letters*, 1:56–58, 1971.
17. Panayiotis J. Psarrakos. On the mth roots of a complex matrix. *The Electronic Journal of Linear Algebra*, 9:32–41, 2002.
18. Kenneth A. Ross and Charles R. B. Wright. *Discrete mathematics*, chapter 7.5. Prentice-Hall, second edition, 1988.
19. Larry L. Stockmeyer. The minimal set basis problem is NP-complete. IBM Research Report No. RC-5431, IBM Thomas J. Watson Research Center, 1975.