Bipartite Roots of Graphs

LAP CHI LAU

University of Toronto

Abstract. Graph H is a root of graph G if there exists a positive integer k such that x and y are adjacent in G if and only if their distance in H is at most k. Motwani and Sudan [1994] proved the NP-completeness of graph square recognition and conjectured that it is also NP-complete to recognize squares of bipartite graphs. The main result of this article is to show that squares of bipartite graphs can be recognized in polynomial time. In fact, we give a polynomial-time algorithm to count the number of different bipartite square roots of a graph, although this number could be exponential in the size of the input graph. By using the ideas developed, we are able to give a new and simpler linear-time algorithm to recognize squares of trees and a new algorithmic proof that tree square roots are unique up to isomorphism. Finally, we prove the NP-completeness of recognizing cubes of bipartite graphs.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.2.2 [**Discrete Mathematics**]: Graph Theory

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Graph roots, graph powers, bipartite graphs

1. *Introduction*

Root and *root finding* are concepts familiar to most branches of mathematics. In graph theory, H is a *root* of $G = (V, E)$ if there exists a positive integer k such that x and y are adjacent in G if and only if their distance in H is at most k. If H is a kth root of G, then we write $G = H^k$ and call G the kth power of H (see Figure 1).

For any class of graphs, recognition is a fundamental structural and algorithmic problem; in this article, we study the recognition problems on graph powers. Ordinarily, it is a difficult task to determine whether a given graph G has a kth root or not. Also, the number of kth roots could be exponential in the size of the input graph. Ross and Harary [1960] characterized squares of trees and showed that tree square roots, when they exist, are unique up to isomorphism. Mukhopadhyay [1967] characterized general graphs that possess a square root, and in the following year, Geller [1968] solved the problem for general digraphs. Escalante et al. [1974] characterized graphs and digraphs with a kth root. However, all characterizations

The author is supported by Microsoft Fellowship.

Author's address: Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ont., M5S 3G4, Canada, e-mail: chi@cs.toronto.edu.

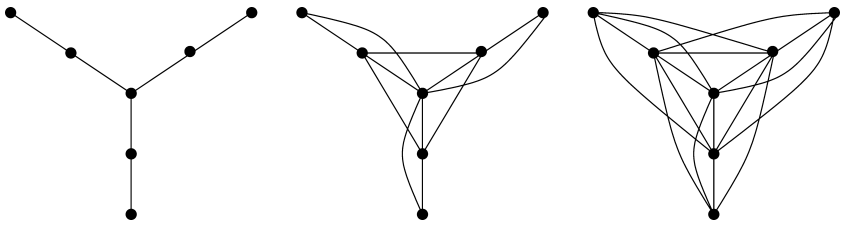

FIG. 1. G (a tree), G^2 and G^3.

of powers of general graphs are not polynomial in the sense that they do not yield polynomial-time algorithms. The complexity of graph power recognition was unresolved until 1994 when Motwani and Sudan [1994] proved the NP-completeness of recognizing squares of graphs and stated that they believed it is also NP-complete to recognize squares of bipartite graphs. About the same time, Lin and Skiena [1995] gave linear-time algorithms for recognizing squares of trees and, based on the characterization given by Harary et al. [1967] finding square roots of planar graphs. Recently, Lau and Corneil [2004] presented polynomial-time algorithms for recognizing kth powers of proper interval graphs for every fixed k. On the other hand, they showed the NP-completeness of recognizing squares of chordal graphs, recognizing squares of split graphs and recognizing chordal graphs that are squares of some graph.

1.1. OUR RESULTS. The main result of this article is to show that squares of bipartite graphs can be recognized in polynomial-time. In Section 2, we give an overview of this algorithm and show how to reduce the problem of recognizing squares of bipartite graphs to two auxiliary problems. The algorithm for the first auxiliary problem is given in Section 3, and an involved algorithm for the second auxiliary problem is presented in Section 4. Then, in Section 5, we show how to modify the algorithm to count the number of different bipartite square roots of a graph. Using the techniques developed, in Section 6, we present a new and simpler linear-time algorithm to recognize square of trees, and a new algorithmic proof that tree square roots are unique up to isomorphism. Finally, in Section 7, we show the NP-completeness of recognizing cubes of bipartite graphs.

1.2. NOTATION AND DEFINITIONS. Our basic notation and terminology reference is West [2001]. We denote a *graph* G with *vertex set* $V(G)$ and *edge set* $E(G)$ by $G = (V, E)$. When u and v are endpoints of an edge, they are *adjacent* and are *neighbors*. We write $u \leftrightarrow v$ or $uv \in E(G)$ for "u is adjacent to v". All the graphs we consider are *simple*, *undirected* and *loopless*, unless otherwise specified. The *complement* $\overline{G}$ of a simple graph G is the simple graph with vertex set $V(G)$ defined by $uv \in E(\overline{G})$ if and only if $uv \notin E(G)$. A graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. G' is an *induced subgraph* of G, written $G[V']$, if it is a subgraph of G and it contains all the edges uv such that $u, v \in V'$ and $uv \in E(G)$.

The *degree* of a vertex v in a graph G, written $deg(v)$ is the number of edges incident with v. The *maximum degree* is denoted $\Delta(G)$. The *open neighborhood* $N_G(v)$ of v is the set of vertices adjacent to v in G, and the *closed neighborhood* $N_G[v]$ of v is $N_G(v) \cup \{v\}$. When U is a set of vertices, $N_G[U] = \bigcup_{v \in U} N_G[v]$. If G has a u,v-path, then the *distance* from u to v, written $d_G(u, v)$ is the least

length of any u, v-path. The kth *neighborhood* $N_G^k(v)$ of v is the set of vertices at distance k from v. A graph G is *connected* if each pair of vertices in G is connected by a path; otherwise, G is *disconnected*. A *maximal connected subgraph* of G is a subgraph that is connected and is not contained in any other connected subgraph. The *components* of a graph are its maximal connected subgraphs.

A *clique* in a graph G is a set of pairwise adjacent vertices. When the set has size r, the clique is denoted by K_r. An *independent set* (or *stable set*) in a graph is a set of pairwise nonadjacent vertices. The *clique number* $\omega(G)$ is the maximum clique size in G; while the *independence number* $\alpha(G)$ is the maximum independent set size in G. The *chromatic number* $\chi(G)$ of a graph G is the minimum number of colors needed to label the vertices so that adjacent vertices receive different colors. The *clique cover number* $\theta(G)$ of a graph G is the minimum number of cliques in G needed to cover $V(G)$. These four parameters of a graph are very important in many optimization problems on graphs.

A graph is *bipartite* if there is a bipartition of its vertex set into two disjoint stable sets called the *partite sets* of G. Two vertices are on the *same side* if they are in the same partite set; otherwise, they are on *different sides*. A *complete bipartite graph* is a bipartite graph such that two vertices are adjacent if and only if they are on different sides. When the sets have size r and s, the complete bipartite graph is denoted by $K_{r,s}$. A *tree* is a connected graph with no cycle. A *leaf* is a vertex v of degree 1; the only neighbor of v is the *parent* of v. A *star* is a tree with diameter 2, and a *double star* is a tree with diameter 3.

2. *Squares of Bipartite Graphs*

In this section, we give an overview of the polynomial-time algorithm to solve the SQUARE OF BIPARTITE GRAPH (SB) problem.

PROBLEM. SQUARE OF BIPARTITE GRAPH (SB)
INSTANCE. A graph G.
QUESTION. Does there exist a *bipartite graph* B such that $B^2 = G$?

Henceforth, we assume, without loss of generality, that G is connected; hence, B must also be connected. The following simple observation is very useful in later proofs.

PROPOSITION 2.1. *Let B be a bipartite graph such that $B^2 = G$. If $uv \in E(G)$ and u, v are on different sides of B, then $uv \in E(B)$.*

PROOF. Suppose, by way of contradiction, that $uv \notin E(B)$. Since u and v are on different sides of B, they do not share any common neighbor in B. So $uv \notin E(B^2)$ and this contradicts the assumption that $B^2 = G$. As a result, $uv \in E(B)$. □

Here we introduce two important auxiliary problems to solve SB. Let $B = (X, Y, E)$ be a bipartite graph with X and Y as the partite sets. Suppose we fix the partite sets of the bipartite roots of G. Then, from Proposition 2.1, the edge set of the bipartite root is *forced*. In fact, the unique bipartite root candidate is $B = (X, Y, E)$ with $E(B) = \{uv \mid uv \in E(G) \text{ and } u \in X \text{ and } v \in Y\}$. Extending this idea, we show that the following problem can be solved in polynomial time.

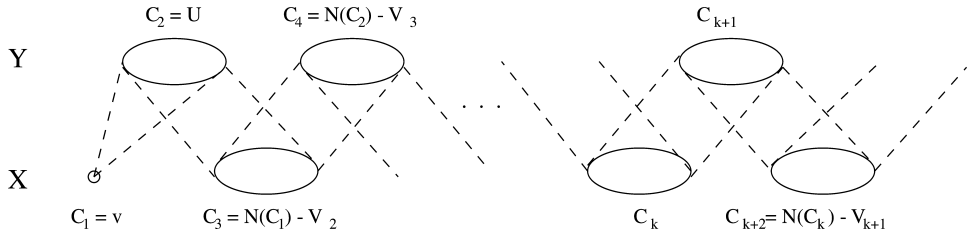

FIG. 2. An illustration of the algorithm SBN.

PROBLEM. SB WITH A SPECIFIED NEIGHBORHOOD (SBN)
INSTANCE. A graph G, $v \in V(G)$ and $U \subseteq N_G(v)$.
QUESTION. Does there exist a bipartite graph B such that $B^2 = G$ and $N_B(v) = U$?

Given an instance of SBN, it turns out that we can construct the unique candidate solution in polynomial time. This result will be presented in Section 3. Now we introduce the second auxiliary problem. We say a vertex v in a bipartite graph is *maximal* if $N_B(v) \not\subset N_B(u)$ for all $u \in V(B)$; an edge $e = uv$ in a bipartite graph is *maximal* if both u and v are maximal vertices. Using the result of SBN, we can further show that the following problem is polynomial-time solvable.

PROBLEM. SB WITH A SPECIFIED MAXIMAL EDGE (SBE)
INSTANCE. A graph G, $xy \in E(G)$.
QUESTION. Does there exist a bipartite graph B such that $B^2 = G$ and xy is a maximal edge in B?

Given an instance of SBE, we can solve it in time $O(M(n))$ by reducing it to at most two instances of SBE, where $M(n)$ is the time complexity of doing a matrix multiplication operation for two $n \times n$ matrices. Here, we need to do matrix multiplications to check if the candidate solutions are actually bipartite square roots of G.

Now, with the polynomial-time algorithm for SBE, we show how to solve SB in polynomial-time. Given an instance of SB, we pick a vertex x in G with maximum degree and note that it must be a maximal vertex in B for any bipartite square root B of G. Also, if G is the square of a bipartite graph B, there is at least one vertex $y \in N_G(x)$ such that $y \in N_B(x)$ and y is maximal in B (consider a vertex y with maximum degree amongst vertices in $N_B(x)$). Therefore, if G is the square of a bipartite graph, then there must exist a neighbor y of x in G with xy being a maximal edge in B. Since $N_G(x)$ is of cardinality at most $\Delta(G)$ where $\Delta(G)$ denotes the maximum degree of G, the time complexity of SB is at most $\Delta(G)$ times the complexity of SBE. This completes the high-level description of the algorithm for SB. The algorithm for SBN is presented in Section 3 and the algorithm for SBE is given in Section 4.

3. *Bipartite Square Roots with One Specified Neighborhood*

As mentioned in the previous section, given an instance of SBN, if there is a solution, then there is a unique solution. The following is an algorithm to find the unique candidate solution. Figure 2 illustrates the concept of the algorithm.

SBN
$C_1 \leftarrow \{v\}$
$C_2 \leftarrow U$
$V_2 \leftarrow C_1 \cup C_2$
$k \leftarrow 2$
while (V_k is a proper subset of $V(G)$) **do**
    $C_{k+1} \leftarrow N_G(C_{k-1}) - V_k$
    $V_{k+1} \leftarrow V_k \cup C_{k+1}$
    $k \leftarrow k + 1$
$X \leftarrow \bigcup_i C_{2i+1}$
$Y \leftarrow \bigcup_i C_{2i}$
$E \leftarrow \{xy \mid x \in X,\ y \in Y$ and $xy \in E(G)\}$

LEMMA 3.1. *Given an instance of* SBN*, the algorithm outputs the unique solution, if some solution exists.*

PROOF. Without loss of generality, we assume $v \in X$. Starting from v in X and U in Y, we enlarge the *forced* bipartition in each iteration. In the algorithm, after the $(k-1)$th iteration, V_k is the set of vertices that are forced to be on one side in B. Since it is a bipartite graph, there is no edge between vertices on the same side. On the other hand, by Proposition 2.1, all the edges between vertices on different sides are in B. So $E_k = \{uw \mid uw \in E(G)$ and $u \in (V_k \cap X)$ and $w \in (V_k \cap Y)\}$ is the forced edge set in $B[V_k]$. Thus, $B_k = (V_k, E_k)$ is the forced induced bipartite subgraph of any bipartite square root B of G with $N_B(v) = U$. Suppose the algorithm terminates at the ith iteration; then $V_i = V(G)$ and $B_i = (V_i, E_i)$ is the forced bipartite graph square root of G.

Explicitly, we will prove by induction on k that if there exists a bipartite graph $B = (X, Y, E)$ such that $B^2 = G$ and $N_B(v) = U$, then B_k is an induced bipartite subgraph of B with the following properties:

(P_{k1})	$C_i \neq \emptyset$	for $1 \leq i \leq k$
(P_{k2})	$u_i \in C_i \Rightarrow N_G(u_i) \subseteq V_k$	for $1 \leq i \leq k-2$
(P_{k3})	for $u_i \in C_i$, $u_i \in X$ iff i is an odd number	for $1 \leq i \leq k$
(P_{k4})	$u_i \in C_i \Rightarrow \exists u_{i-1} \in C_{i-1}$ such that $u_{i-1}u_i \in E_k$	for $2 \leq i \leq k$

Without loss of generality, we assume $v \cup U \subset V(G)$ and $U \neq \emptyset$. By the specification of SBN, B_2 is clearly an induced bipartite subgraph of any bipartite square root B of G with $N_B(v) = U$. Since (P_{k2}) is only applicable when $k \geq 3$, the base case is B_3. We will show that if there exists a bipartite graph B such that $B^2 = G$ and $N_B(v) = U$, then (P_{31})-(P_{34}) hold.

For (P_{31}), $C_1 = \{v\}$ and $C_2 = U \subseteq N_G(v)$ are not empty by our assumption. Suppose, by way of contradiction, that $C_3 = \emptyset$. From the algorithm, $C_3 = N_G(C_1) - V_2$. Since $C_3 = \emptyset$, we have $N_G(v) = C_2 = U = N_B(v)$. But then, it is easy to see that if there is an edge between V_2 and $V(B) - V_2$ in B, then $B^2 \neq G$. So V_2 is disconnected from $V(B) - V_2$ in any bipartite square root B of G such that $N_B(v) = U$. Since $V_2 \subset V(B)$, B is disconnected and this contradicts the assumption that G is connected. So (P_{31}) holds in B_3.

By the algorithm, $C_3 = N_G(C_1) - V_2$. So $N_G(v) = C_2 \cup C_3 \subseteq V_3$ and thus (P_{32}) holds.

For (P_{33}), $C_1 = \{v\}$ is in X by assumption. Since $C_2 = U = N_B(v)$, C_2 must be in Y. By the algorithm, $C_3 = N_G(v) - V_2$ and thus $C_3 = N_G(v) - N_B[v]$. By (P_{31}),

$C_3 \neq \emptyset$. So any vertex u in C_3 is adjacent to v in G but not in B. By Proposition 2.1, C_3 must be in X and thus (P_{33}) holds.

For (P_{34}), it is true for vertices in C_2 since $C_2 = U = N_B(v)$. For any vertex $u_3 \in C_3$, by (P_{33}), it is on the same side as v in B. Also it is adjacent to v in G. So, if $B^2 = G$, u_3 and v should share at least one common neighbor in B. Since $N_B(v) = C_2$ and $C_2 \neq \emptyset$, u_3 has at least one neighbor in C_2 and thus (P_{34}) holds. So the base case holds and therefore B_3 is an induced bipartite subgraph for any bipartite square root B of G with $N_B(v) = U$.

Now assume the statements hold for B_k for $k \geq 3$; B_k is an induced bipartite subgraph for any bipartite square root B of G such that $N_B(v) = U$. If $V_k = V(G)$, then we are done. So we assume $V_k \subset V(G)$. We show that the statements also hold for B_{k+1}. In the following, we assume that k is even. The case when k is odd uses exactly the same arguments.

For ($P_{k+1,1}$), by the induction hypothesis, (P_{k1}) holds and so $C_1, \ldots, C_k$ are not empty. It suffices to show that $C_{k+1} \neq \emptyset$. Suppose, by way of contradiction, that $C_{k+1} = \emptyset$. Then $V_{k+1} = V_k$ and so $V_{k+1} \subset V(G)$. By the induction hypothesis, (P_{k2}) holds in B_k and so $N_G(V_{k-2}) \subseteq V_k$. From the algorithm, $C_{k+1} = N_G(C_{k-1}) - V_k$. Since $C_{k+1} = \emptyset$, it implies $N_G(V_{k-1}) \subseteq V_k$. Now we will argue that $V_{k+1} = V_k$ is disconnected from $V(B) - V_{k+1}$ in any bipartite square root B of G such that $N_B(v) = U$. Since $N_G(V_{k-1}) \subseteq V_k$, if $B^2 = G$, then $N_B(V_{k-1}) \subseteq N_G(V_{k-1})$ and thus there is no edge between V_{k-1} and $V(G) - V_k$ in B. The only possibility left is that there is an edge $u_k v$ between $u_k \in C_k$ and $v \in V(G) - V_k$ in B. By the induction hypothesis, B_k is an induced subgraph of B with (P_{k1}-P_{k4}) hold. By (P_{k4}) of B_k, $u_k \in C_k$ implies there exists $u_{k-1} \in C_{k-1}$ such that $u_k u_{k-1}$ in E_k and thus in $E(B)$. However, this implies that $v u_{k-1} \in E(B^2)$ and this contradicts the assumption that $B^2 = G$. Therefore, V_{k+1} is disconnected in any bipartite square root B of G such that $N_B(v) = U$. But this contradicts the assumption that G is connected. So ($P_{k+1,1}$) holds in B_{k+1}.

By the induction hypothesis, (P_{k2}) holds in H_k. By the algorithm, $C_{k+1} = N_G(C_{k-1}) - V_k$ and so $N_G(C_{k-1}) \subseteq V_{k+1}$, thus ($P_{k+1,2}$) holds in B_{k+1}.

Now we consider ($P_{k+1,3}$). Since we assume k is even, by the induction hypothesis (P_{k3}), C_{k-1} is in X. By ($P_{k+1,1}$), $C_{k+1} \neq \emptyset$. For any vertex u_{k+1} in C_{k+1}, it is adjacent to a vertex u_{k-1} in C_{k-1} in G. Suppose, by way of contradiction, that u_{k+1} is in Y. By Proposition 2.1, $u_{k+1}u_{k-1} \in E(B)$. By the induction hypothesis, B_k is an induced subgraph of B with (P_{k1}-P_{k4}) hold. By (P_{k4}) of B_k, $u_{k-1} \in C_{k-1}$ implies there exists $u_{k-2} \in C_{k-2}$ such that $u_{k-1}u_{k-2}$ in E_k and thus in $E(B)$. However, this implies that $u_{k+1}u_{k-2} \in E(B^2)$ but this contradicts the assumption that $B^2 = G$. So ($P_{k+1,3}$) holds in B_{k+1}.

For ($P_{k+1,4}$), by the induction hypothesis (P_{k4}), it suffices to show that it is also true for vertices in C_{k+1}. For any vertex u_{k+1} in C_{k+1}, it is adjacent to a vertex u_{k-1} in C_{k-1} in G. By ($P_{k+1,3}$), C_{k+1} and C_{k-1} are on the same side. Therefore, if $B^2 = G$, u_{k+1}, u_{k-1} must share a common neighbor in B. By ($P_{k,2}$), $N_G(V_{k-2}) \subseteq V_k$. Since $B^2 = G$, $N_G(V_{k-2}) \subseteq V_k$ implies $N_B(V_{k-2}) \subseteq V_k$. So, u_{k+1} does not have a neighbor in V_{k-2} in B. By ($P_{k+1,2}$), $N_G(C_{k-1}) \subseteq V_{k+1}$. So, for u_{k+1} to share a common neighbor with u_{k-1}, u_{k+1} must have a neighbor in C_k. So ($P_{k+1,4}$) 2 holds.

As a result, B_{k+1} is an induced bipartite subgraph of any bipartite square root B of G such that $N_B(v) = U$. By (P_{k1}), in each iteration, we add at least one more vertex to the forced bipartition of B. Eventually, there is an $i \leq n$ such that $V_i = V(G)$ and then $B = B_i$ is the unique bipartite square root of G such that $N_B(v) = G$, if some solution exists. □

LEMMA 3.2. SBN *can be solved in time* $\mathcal{O}(M(n))$.

PROOF. We solve this problem in two phases. In the first phase we construct the unique candidate B as shown in Lemma 3.1. By using an implementation similar to breadth first search, B can be constructed in linear-time. Notice that even if G does not have a square root, B is constructed. So in the second phase we have to check if $B^2 = G$. If $B^2 = G$, we say "YES". Otherwise, we say "NO" since B is the unique candidate. The second phase can be done by doing matrix multiplication and thus the total complexity is $\mathcal{O}(M(n))$. □

4. *Bipartite Square Roots with One Specified Maximal Edge*

In this section, we present a polynomial-time algorithm for SBE. In particular, we reduce SBE to at most 2 instances of SBN. Given the partial solution where initially x and y are on different sides, the algorithm incrementally enlarges the partial (forced) solution until it can be reduced to at most 2 instances of SBN. The organization of this section is as follows. In Section 4.1, we first introduce the necessary notations and prove some preliminary structural results. Then we outline the algorithm in Section 4.2. In Section 4.3, we give the algorithmic details and prove the correctness of each step.

4.1. PRELIMINARIES. Recall that the chromatic number $\chi(G)$ is the minimum number of colors needed to label the vertices of G so that adjacent vertices receive different colors. It is well known that a graph G is bipartite if and only if $\chi(G) \leq 2$. The clique covering number $\theta(G)$ of a graph G is the minimum number of cliques in G to cover $V(G)$; note that $\theta(G) = \chi(\overline{G})$. So we say that a graph G with $\theta(G) \leq 2$ a *co-bipartite graph* (i.e., the complement of a bipartite graph).

Given a co-bipartite graph H; we denote the connected components of $\overline{H}$ by $\overline{C}_1, \ldots, \overline{C}_k$ and we say they are the *co-components* of H (i.e., connected components in the complement of H). A co-component is *trivial* if its vertex set is of size 1. For any co-component $\overline{C}_i$ of a co-bipartite graph H, $\overline{H}[\overline{C}_i]$ is a connected bipartite graph, we call the vertex set that corresponds to a partite set of $\overline{H}[\overline{C}_i]$ a *part*. Let H be an induced co-bipartite subgraph of G and let $\overline{C}_1, \ldots, \overline{C}_k$ be its co-components. A vertex $v \in V(G) - V(H)$ is of:

—*type* 0 to $\overline{C}_i$ if v is not adjacent to any vertex in $\overline{C}_i$;

—*type* 1 to $\overline{C}_i$ if v is adjacent to some vertices in exactly one part, called *adjacent part*, of $\overline{C}_i$;

—*type* 2 to $\overline{C}_i$ if v is adjacent to some vertices in both parts of $\overline{C}_i$.

A vertex v is *type* 1 *universal* to $\overline{C}_i$ if it is adjacent to all vertices in the adjacent part; similarly, v is *type* 2 *universal* to $\overline{C}_i$ if it is adjacent to all vertices in $\overline{C}_i$.

Suppose x and y are the vertices that are part of the input to SBE. We always assume $x \in X$ and $y \in Y$ in B. Let $P_x = N_G(x) - N_G(y)$, $P_y = N_G(y) - N_G(x)$ and we say $P_{xy} = P_x \cup P_y$ is the set of *private neighbors*. Also, we say $C_{xy} = N_G(x) \cap N_G(y)$ is the set of *common neighbors*. The following lemma describes the structures of the vertices in C_{xy} in B and G.

LEMMA 4.1. *Suppose* $B = (X, Y, E)$, $xy \in E(B)$ *and* $B^2 = G$;
then $(N_B(x) - y) \cup (N_B(y) - x) = C_{xy}$ *and* $G[C_{xy}]$ *is a co-bipartite graph.*

PROOF. First, we show that $(N_B(x) - y) \cup (N_B(y) - x) = C_{xy}$. Consider an arbitrary vertex $v \in C_{xy}$; we claim that v is either in $N_B(x)$ or $N_B(y)$. Suppose v is in X; since $vy \in E(G)$, by Proposition 2.1, $vy \in E(B)$. A similar argument applies if v is in Y. Furthermore, since we consider graphs without loops, x and y are not in C_{xy}. So we have $C_{xy} \subseteq (N_B(x) - y) \cup (N_B(y) - x)$. On the other hand, consider an arbitrary vertex $u \in N_B(x) - y$, it is obvious that $u \in N_{B^2}(x)$. Also, since $xy \in E(B)$, $u \in N_{B^2}(y)$. A similar argument applies if $u \in N_B(y) - x$. Therefore, $u \in N_{B^2}(x) \cap N_{B^2}(y)$. Since $B^2 = G$, $u \in C_{xy}$. Hence, $(N_B(x) - y) \cup (N_B(y) - x) = C_{xy}$. Finally, $N_B(x) - y$ and $N_B(y) - x$ induce cliques in B^2 and thus in G. So $G[C_{xy}]$ is a co-bipartite graph. □

By Lemma 4.1, $G[C_{xy}]$ is a co-bipartite graph. Henceforth, we let the trivial co-components of $G[C_{xy}]$ be $\overline{c_1}, \ldots, \overline{c_l}$ and the co-components be $\overline{C}_1, \ldots, \overline{C}_k$. Before we proceed to the outline of the algorithm, we first show an important lemma of placing co-components, which will be used implicitly many times (see Section 4.2).

LEMMA 4.2. *Suppose $B = (X, Y, E)$, $xy \in E(B)$ and $B^2 = G$. Given a co-component $\overline{C}$ of $G[C_{xy}]$; all vertices in one part of $\overline{C}$ must be on one side of B while all vertices in the other part of $\overline{C}$ must be on the other side of B.*

PROOF. Let U and W be the two parts of $\overline{C}$ and $U = U_1 \cup U_2$ and $W = W_1 \cup W_2$. Suppose that, in B, U_1 and W_1 are in X while U_2 and W_2 are in Y. Since U_1 and W_1 are in X, by Proposition 2.1, y is adjacent to every vertex in U_1 and W_1 in B and thus $G[U_1 \cup W_1]$ is a clique. By the same argument, $G[U_2 \cup W_2]$ is a clique. Also $G[U]$ and $G[W]$ are cliques by the definition of a part of a co-component. So in G we have all possible edges between $\{U_1 \cup W_2\}$ and $\{U_2 \cup W_1\}$ and thus they are disconnected in $\overline{G}$. Now we have contradicted the fact that $\overline{C}$ is a co-component unless one of $\{U_1 \cup W_2\}$ or $\{U_2 \cup W_1\}$ is empty and thus the lemma holds. □

4.2. OUTLINE OF THE ALGORITHM. By Lemma 4.2, given a co-component $\overline{C} = \{U \cup W\}$ of $G[X_{xy}]$ where U and W are the two parts of $\overline{C}$, we just have to consider the *orientation* of $\overline{C}$ in B (i.e., whether U is placed in X and W is placed in Y, or U is placed in Y and W is placed in X). If the orientation of a co-component has been fixed (i.e., it was placed by the algorithm), we refer to it as a *fixed co-component*; otherwise, it is a *free co-component*. At the beginning of the algorithm, every co-component is free.

Our algorithm will place $N_G(x) \cup N_G(y)$ in B in several steps; in each step some vertices will be placed in B. In the first step, all vertices in P_{xy} of G will be placed in B. Then, all trivial co-components of $G[C_{xy}]$ will be placed in B in the second step. Note that if there is just one free co-component $\overline{C}$ left, then we are finished since we just have to consider at most two possibilities by Lemma 4.2, and each instance can be solved by a call to SBN. The difficult case is when we have many free co-components. So, henceforth, when we place co-components in B, we make the following assumption.

Remark 4.3. There are at least two free co-components left when we place co-components in B.

Using the positions of vertices in P_{xy} in B and the edges in G between vertices in P_{xy} and the vertices in free co-components, we will fix the free co-components in B in many different steps. In each step, we look for some configurations in G from which we can force the orientations of some free co-components in B. Then after we fix those free co-components, we can assume more structures in the remaining free co-components, and we can force the orientations of more remaining free co-components in B. After several such procedures, if there are still at least two free co-components left, we have a very symmetrical structure where the remaining free co-components are very similar to each other. To finish up, we argue that any arbitrary orientations of the remaining free co-components would give the same squared graph. Therefore, we can just place the remaining free co-components arbitrarily, and the problem reduces to an instance of SBN, and we are done by using the result in Section 3. The following is an elaborated algorithm outline, this is intended to be used by the reader for future reference.

SBE

1 PLACE PRIVATE NEIGHBORS
Description: Place P_x in X and P_y in Y.

2 CHECK CO-BIPARTITENESS
Description: Ensure that $G[C_{xy}]$ is a co-bipartite graph.

3 PLACE TRIVIAL CO-COMPONENTS
Description: Place all trivial co-components.
Post-condition (3.1) (by Corollary 4.8): In B, there is no edge between any trivial co-component and vertices in P_{xy}.

4 NO TYPE 0 VERTEX
Assumption: (3.1)
Description: Ensure that (4.1) happens.
Post-condition (4.1) (by Corollary 4.11): For $v \in P_{xy}$, v is not a type 0 vertex to any co-component.

5 TYPE 2 FORCING
Assumption: (4.1)
Description: If $v \in P_{xy}$ is a type 2 vertex to a co-component, then we place all co-components such that v is not a universal type 2 vertex to them.
Post-condition (5.1) (by Corollary 4.14): If $v \in P_{xy}$ is a type 2 vertex to a co-component in G, then v is a universal type 2 vertex to all free co-components in G.
Post-condition (5.2) (by Corollary 4.15): If $v \in P_{xy}$ is a type 1 vertex to a free co-component in G, then v is a type 1 vertex to all co-components in G.

6 TYPE 1 FORCING
Assumption: (3.1), (5.2)
Description: We place all co-components *unless* (6.1) happens.
Post-condition (6.1) (by Corollary 4.18): For all $v \in P_{xy}$, if v is a type 1 vertex to a free co-component in G, then v is a universal type 1 vertex to all co-components and v's adjacent part of all fixed co-components were placed on the same side as v.

7 MORE TYPE 1 FORCING
Assumption: (5.2)
Description: If there exist type 1 vertices in P_{xy}, then we place all co-components *unless* (7.1) happens.
Post-condition (7.1) (by Corollary 4.20): Type 1 vertices in P_{xy} that are on the same side have the same neighborhood on the free co-components and type 1 vertices in P_{xy} that are on different sides have disjoint neighborhood on the free co-components.

8 PLACE INCOMPLETE CO-COMPONENT
Assumption: (5.2), (6.1)
Description: If $v \in P_{xy}$ is a type 1 vertex, then we place all incomplete co-components.
Post-condition (8.1) (by Corollary 4.22): If there exists a type 1 vertex in P_{xy}, then every free co-component is a complete co-component.
9 FINAL PLACEMENT I
Assumptions: (4.1), (5.1), (5.2), (6.1), (7.1), (8.1).
Description: Place all co-components if $V(G) = N_G(x) \cup N_G(y)$.
10 FINAL PLACEMENT II
Assumptions: (4.1), (5.1), (5.2), (6.1).
Description: Place all co-components if $V(G) \subset N_G(x) \cup N_G(y)$.

4.3. ALGORITHM. Notice that the algorithm is executed in a sequential order. If at any point the **return** command is executed, the output is returned and the execution halts. Also, if the **assert** command fails, the execution of the algorithm will terminate and return "NO". So when we analyze properties of the graph before the execution of a particular step, we assume the previous steps are finished and the execution has not halted yet.

PLACE PRIVATE NEIGHBORS
(1) **for** any vertex $x' \in P_x$
place x' in X
(2) **for** any vertex $y' \in P_y$
place y' in Y

4.3.1. *Place Private Neighbors.* The following lemma shows the correctness of PLACE PRIVATE NEIGHBORS.

LEMMA 4.4. *Suppose $B = (X, Y, E)$, $xy \in E(B)$ and $B^2 = G$; then, in B, every vertex in P_x is in X and every vertex in P_y is in Y.*

PROOF. Suppose, by way of contradiction, that $x' \in P_x$ but x' is placed in Y in B. So, x and x' are on different sides of B. Since $xx' \in E(G)$, by Proposition 2.1, $xx' \in E(B)$. Since $xy \in E(B)$ by assumption, $x'y \in E(B^2)$. On the other hand, since $x' \in P_x$, $x'y \notin E(G)$. Therefore, $B^2 \neq G$; a contradiction. The same argument applies for $y' \in P_y$; thus, the lemma holds. □

4.3.2. *Check Co-Bipartiteness.* Before we proceed, by Lemma 4.1, we first ensure that $G[C_{xy}]$ is a co-bipartite graph.

CHECK CO-BIPARTITENESS
(1) compute $\overline{G}[C_{xy}]$
(2) **assert** that each connected component of $\overline{G}[C_{xy}]$ is a bipartite graph

4.3.3. *Place Trivial Co-Components.* Recall that a trivial co-component $\overline{c}$ of $G[C_{xy}]$ is an isolated vertex in $\overline{G}[C_{xy}]$ (i.e., in G, $\overline{c}$ is adjacent to every vertex in $C_{xy} - \overline{c}$). The following is the algorithm to place trivial co-components:

PLACE TRIVIAL CO-COMPONENTS

Case 1. $P_{xy} \neq \emptyset$
  for any trivial co-component $\overline{c}$
    assert that either $N_G[\overline{c}] = N_G[x]$ or $N_G[\overline{c}] = N_G[y]$
    if $N_G[\overline{c}] = N_G[x]$
    then place $\overline{c}$ in X
    else place $\overline{c}$ in Y
Case 2. $P_{xy} = \emptyset$
  for any trivial co-component $\overline{c}$
    place $\overline{c}$ *arbitrarily* in X or Y

To prove the correctness of PLACE TRIVIAL CO-COMPONENTS, we need the following two lemmas.

LEMMA 4.5. *Suppose $B = (X, Y, E)$, $xy \in E(B)$ and $B^2 = G$. Let $\overline{c}$ be a trivial co-component of $G[C_{xy}]$ in G. If $\overline{c} \in X$, then $N_B[x] \subseteq N_B[\overline{c}]$ and $N_G[x] \subseteq N_G[\overline{c}]$. Otherwise, if $\overline{c} \in Y$, then $N_B[y] \subseteq N_B[\overline{c}]$ and $N_G[y] \subseteq N_G[\overline{c}]$.*

PROOF. By Lemma 4.1, $(N_B(x) - y) \cup (N_B(y) - x) = C_{xy}$. Since $\overline{c}$ is a trivial co-component of $G[C_{xy}]$, $\overline{c}$ is adjacent to all other vertices in C_{xy} in G. If $\overline{c} \in X$ in B, it is in a different partite set than $N_B(x)$ in B. Since $N_B(x) - y \subseteq C_{xy}$, by Proposition 2.1, $\overline{c}$ is adjacent to all vertices in $N_B(x) - y$ in B. Also, since $\overline{c}y \in E(G)$, by Proposition 2.1, $\overline{c}y \in E(B)$. Therefore, $N_B(x) \subseteq N_B(\overline{c})$. Since $B^2 = G$, $N_G(x) \subseteq N_G(\overline{c})$. By the same argument, if $\overline{c} \in Y$ in B, then $N_B(y) \subseteq N_B(\overline{c})$ and $N_G(y) \subseteq N_G(\overline{c})$. □

LEMMA 4.6. *Suppose $B = (X, Y, E)$ and $B^2 = G$. Let u and v be vertices in different partite sets of B. Then $N_G[u] = N_G[v]$ if and only if u and v are both universal vertices in B.*

PROOF. One direction is easy; if u and v are both universal vertices in B, since $B^2 = G$, $N_{B^2}[u] = N_{B^2}[v] = N_G[u] = N_G[v] = V(G)$.

Now we prove the other direction. Suppose, by way of contradiction, that u and v are vertices in different partite sets of B and $N_G[u] = N_G[v]$ but u and v are not both universal vertices in B. Let $S = N_B[u] \cup N_B[v]$ and $T = V(B) - S$. Since u and v are not both universal vertices in B, $T \neq \emptyset$. We now show that S and T are disconnected in B. Suppose, by way of contradiction, that there is an edge $st \in E(B)$ such that $s \in S$ and $t \in T$. Without loss of generality, we assume s is on the same side as u and thus on the different side than v. By our construction of S, $s \in N_B(v)$. So $vt \in E(B^2)$. On the other hand, $t \notin S$ implies $t \notin N_B(u)$. Since u and t are on different sides of B, $ut \notin E(B^2)$. Since $B^2 = G$, $N_G[u] \neq N_G[v]$; a contradiction. So S and T are disconnected in B; however, this contradicts the assumption that G is connected. We conclude that u and v are both universal vertices in B. □

Now we are ready to prove the correctness of PLACE TRIVIAL CO-COMPONENTS.

LEMMA 4.7. PLACE TRIVIAL CO-COMPONENTS *is correct.*

PROOF

Case 1. $P_{xy} \neq \emptyset$. By Lemma 4.4, in any bipartite graph B such that $B^2 = G$ and $xy \in E(B)$, there is no edge between $\{x, y\}$ and P_{xy} in B. Since $P_{xy} \neq \emptyset$, x and y are not both universal vertices in B. First, we show that if $N_G[\overline{c}] \neq N_G[x]$,

then $\overline{c}$ can not be placed in X. If $N_G[\overline{c}] \neq N_G[x]$, then either $N_G[x] \not\subseteq N_G[\overline{c}]$ or $N_G[x] \subset N_G[\overline{c}]$. In the former case, $\overline{c}$ can not be placed in X by Lemma 4.5; in the latter case, $\overline{c}$ can not be placed in X by the maximality of x (recall that, in SBE, we require x to be a maximal vertex in B).

Now we show that if $N_G[x] = N_G[\overline{c}]$, then $\overline{c}$ must be placed in X in B. Suppose, by way of contradiction, that $\overline{c}$ is placed in Y in B. By Lemma 4.6, $\overline{c}$ and x are both universal vertices in B. Since x is a universal vertex in B, $N_G(x) = V(G)$ and thus $P_y = \emptyset$. Since $P_{xy} \neq \emptyset$, $P_x \neq \emptyset$ and thus y is not a universal vertex in B. So $N_B(y) \subset N_B(\overline{c})$; this contradicts the assumption that y is a maximal vertex in B and thus B is not a solution to SBE. The same argument applies when x is replaced by y. Also notice that since $P_{xy} \neq \emptyset$, by Lemma 4.6, $N_G[x] \neq N_G[y]$ and thus there is no ambiguity in the Case 1 of this algorithm. Hence, for any trivial co-component $\overline{c}$, the position of $\overline{c}$ is well defined and thus Case 1 is correct.

Case 2. $P_{xy} = \emptyset$. Let B be a bipartite graph with $\overline{c}$ is in X in B. Let B' be a bipartite graph with the same vertex partition as B except $\overline{c}$ is in Y in B'. By Proposition 2.1, if $B^2 = G$, then $E(B) = \{uv \mid u \in X,\ v \in Y \text{ and } uv \in E(G)\}$ and similarly for $E(B')$. We will show that $E(B'^2) = E(B^2)$. Since $P_{xy} = \emptyset$, by Lemma 4.6, x and y are both universal vertices in B and B'. For any two vertices x_1, x_2 in $X - \overline{c}$; since y is universal in B and B', $x_1x_2 \in E(B^2)$ and also $x_1x_2 \in E(B'^2)$. The same argument applies for any two vertices y_1, y_2 in $Y - \overline{c}$. For any two vertices $x_1 \in X - \overline{c}$, $y_1 \in Y - \overline{c}$; the adjacency is the same in B^2 and B'^2. Note that since $V(G) = \{x\} \cup \{y\} \cup C_{xy}$ and $\overline{c}$ is a trivial co-component in C_{xy}, $\overline{c}$ must be a universal vertex in B (i.e., $N_B(\overline{c}) = Y$) and B' (i.e., $N_{B'}(\overline{c}) = X$). So, $N_{B^2}[\overline{c}] = N_{B'^2}[\overline{c}] = V(G)$. Therefore $E(B^2) = E(B'^2)$; to construct a bipartite square root of G, we can place $\overline{c}$ arbitrarily. Hence, Case 2 of PLACE TRIVIAL CO-COMPONENTS is also correct. □

COROLLARY 4.8. *In B, there is no edge between any trivial co-component $\overline{c}$ of $G[C_{xy}]$ and P_{xy}.*

PROOF. For any two vertices u, v on the same side in B, if $N_B(u) \neq N_B(v)$, then $N_{B^2}[u] \neq N_{B^2}[v]$. For any trivial co-component $\overline{c}$ of $G[C_{xy}]$, by the assertion in PLACE TRIVIAL CO-COMPONENTS, $N_G[\overline{c}]$ is either equal to $N_G[x]$ or $N_G[y]$. Since $B^2 = G$, $N_B(\overline{c})$ is either equal to $N_B(x)$ or $N_B(y)$. It is clear that x has no edge to P_y and y has no edge to P_x in B by the definition of private neighbor. Also, by Lemma 4.4, x and P_x are on the same side in B and similarly for y and P_y. So, $\{x, y\}$ has no edge to P_{xy} in B and thus there is no edge between $\overline{c}$ and P_{xy} in B. □

4.3.4. *No Type* 0 *Vertex.* We now show that for any vertex $v \in P_{xy}$, it is either a type 1 or a type 2 vertex to any co-component $\overline{C}$.

NO TYPE 0 VERTEX

for $v \in P_{xy}$
  assert that v is not a type 0 vertex to some co-component

LEMMA 4.9. *Suppose $B = (X, Y, E)$, $xy \in E(B)$, $B^2 = G$ and $v \in P_{xy}$ is adjacent to $z \in C_{xy}$ in B. If $\overline{C}$ is a co-component such that $z \notin \overline{C}$ and there is a vertex $u \in \overline{C}$ but $uv \notin E(G)$, then $\overline{C}$ can not be placed such that u and v are on the same side of B.*

PROOF. Suppose, by way of contradiction, that u and v are on the same side of B. Since z is in a different co-component than $\overline{C}$, $zu \in E(G)$. Since $zu, zv \in E(G)$, by Proposition 2.1, $zu, zv \in E(B)$. Hence, $uv \in E(B^2)$ but this contradicts the assumption that $B^2 = G$. □

LEMMA 4.10. NO TYPE 0 VERTEX *is correct.*

PROOF. Suppose, by way of contradiction, that v is a type 0 vertex to a co-component $\overline{C}$ in G. Without loss of generality, we assume that $v \in P_x$ in B. By PLACE PRIVATE NEIGHBORS, v is placed in X. By Lemma 4.1, $N_B(x) - y \subseteq C_{xy}$. By Corollary 4.8, v is not adjacent to any trivial co-component in B. Since $v \in P_x$, v and x must have a common neighbor z in B (clearly, z is placed in Y). So z is in a co-component $\overline{C}'$ such that $\overline{C}' \neq \overline{C}$. Since $\overline{C}$ is a co-component, there are vertices u, w on different parts of $\overline{C}$ that are not adjacent to v in G. Since $vz \in E(B)$ and $z \notin \overline{C}$, by Lemma 4.9, neither u nor w can be placed on the same side as z or otherwise the assumption that $B^2 = G$ is contradicted. Therefore, if $B^2 = G$, v does not exist and the lemma holds. □

COROLLARY 4.11. *After the execution of* NO TYPE 0 VERTEX*, for any vertex* $v \in P_{xy}$ *and any co-component* $\overline{C}$*,* v *is either a type* 1 *vertex to* $\overline{C}$ *or a type* 2 *vertex to* $\overline{C}$.

4.3.5. *Type* 2 *Forcing.* The following algorithm forces the orientation of some free co-components based on the position of vertices in P_{xy} in B which are type 2 vertices to some co-components in G:

```
TYPE 2 FORCING
for v ∈ P_xy
    if v is a type 2 vertex to some co-component C̄'
      1. for any free co-component C̄ ≠ C̄'
             if there is a vertex u ∈ C̄ such that uv ∉ E(G)
                 place C̄ such that u is on the side opposite v
      2. if there are at least two free co-components left
             if there is a vertex u ∈ C̄' such that uv ∉ E(G)
                 place C̄' such that u is on the side opposite v
```

LEMMA 4.12. *Suppose* $B = (X, Y, E)$*,* $xy \in E(B)$*,* $B^2 = G$ *and* $v \in P_{xy}$ *is a type* 2 *vertex to a co-component* $\overline{C}'$*. If* $\overline{C}$ *is a co-component such that* $\overline{C} \neq \overline{C}'$ *and there is a vertex* $u \in \overline{C}$ *that is not adjacent to* v*, then* $\overline{C}$ *can not be placed such that* u *and* v *are on the same side of* B.

PROOF. Let w and z be vertices of different parts of $\overline{C}'$ such that they are adjacent to v. By Lemma 4.2, exactly one of w or z is on the side opposite v in B. Without loss of generality, we assume z is on the side opposite v in B. By Proposition 2.1, $zv \in E(B)$. Hence, by Lemma 4.9, the lemma follows. □

LEMMA 4.13. TYPE 2 FORCING *is correct.*

PROOF. The correctness of the first part of the algorithm is justified by Lemma 4.12. Suppose there are two free co-components left after the first part

(note that one is $\overline{C}'$); there is a free co-component $\overline{C}$ such that $\overline{C} \neq \overline{C}'$. Since $\overline{C}$ is not fixed by the first part of TYPE 2 FORCING, v must be a universal type 2 vertex to $\overline{C}$. So, by applying Lemma 4.12 with $\overline{C}'$ replaced by $\overline{C}$, we prove the correctness of the second part of TYPE 2 FORCING. □

COROLLARY 4.14. *After the execution of* TYPE 2 FORCING, *if v is a type 2 vertex to a co-component in G and there are at least two free co-components left, then v is a universal type 2 vertex to all free co-components.*

PROOF. This follows from the first part of TYPE 2 FORCING. □

COROLLARY 4.15. *After the execution of* TYPE 2 FORCING, *if v is a type 1 vertex to a free co-component in G, then v is a type 1 vertex to all co-components in G.*

PROOF. By Remark 4.3, there are two free co-components left. By Corollary 4.14, if v is a type 2 vertex to some co-component, then v is a universal type 2 vertex to all free co-components. So, if v is a type 1 vertex to a free co-component, v is not a type 2 vertex to any co-component. Also, by Lemma 4.10, v is not a type 0 vertex to any co-component. So, if v is a type 1 vertex to a free co-component $\overline{C}$, v is a type 1 vertex to all co-components. □

4.3.6. *Type 1 Forcing.* In TYPE 1 FORCING, we use the vertices in P_{xy} that are type 1 to some co-components to fix the orientations of some free co-components.

LEMMA 4.16. *Suppose $B = (X, Y, E)$, $xy \in E(B)$ and $B^2 = G$. If $v \in P_{xy}$ is a type 1 vertex to all co-components in G, there is exactly one co-component $\overline{C'}$ such that the adjacent part of $\overline{C'}$ to v is on the side opposite v in B. Furthermore, v is a universal type 1 vertex to any co-component $\overline{C}$ in G where $\overline{C} \neq \overline{C'}$.*

PROOF. First, we prove that there is at least one co-component in B such that the adjacent part is on the side opposite v. Suppose, by way of contradiction, that there is no co-component such that the adjacent part is on the side opposite v in B. Without loss of generality, we assume that $v \in P_x$. Since v is not adjacent to any trivial co-component in B by Corollary 4.8, v does not share a common neighbor with x in B. Hence, $vx \notin E(B^2)$; this contradicts the assumption that $v \in P_x$.

Now we prove that there is exactly one co-component $\overline{C'}$ such that the adjacent part is on the side opposite v, and for other co-components, v is a universal type 1 vertex to them. First, there exists a vertex $z \in \overline{C'}$ such that $zv \in E(G)$ and z and v are on different sides of B. By Proposition 2.1, $zv \in E(B)$. Suppose, by way of contradiction, that there exists $\overline{C} \neq \overline{C'}$ that u, w are in different parts of $\overline{C}$ but not adjacent to v. Since $zv \in E(B)$ and $z \notin \overline{C}$, by Lemma 4.9, neither u nor w can be placed on the same side as v or otherwise the assumption that $B^2 = G$ is contradicted. Therefore, if $\overline{C} \neq \overline{C'}$, then v must be a universal type 1 vertex to $\overline{C}$. By the same argument (i.e., using Lemma 4.9), the adjacent part of $\overline{C}$ must be placed on the same side as v. □

Let $v \in P_{xy}$ be a type 1 vertex to all co-components in G. By Lemma 4.16, if $xy \in E(B)$ and $B^2 = G$, then there is exactly one co-component $\overline{C}$ such that the adjacent part of $\overline{C}$ to v is on the side opposite v in B; we call $\overline{C}$ the *active co-component* of v in B.

TYPE 1 FORCING
for $v \in P_{xy}$
if v is a type 1 vertex to some free co-component

1. **if** there exists a fixed co-component $\overline{C}'$ such that
$\overline{C}'$'s adjacent part to v is on the side opposite v
for any free co-component $\overline{C}$
place $\overline{C}$ such that the adjacent part to v
is on the same side with v
return SBN
2. **assert** that v is a type 1 universal vertex
to every fixed co-component
3. **if** there is one free co-component $\overline{C}$ such that
v is not a universal type 1 vertex to $\overline{C}$
place $\overline{C}$ s.t. $\overline{C}$'s adj. part to v is on the side opposite v
for any free co-component $\overline{C}' \neq \overline{C}$
place $\overline{C}'$ such that $\overline{C}'$'s adjacent part to v
is on the same side with v
return SBN

LEMMA 4.17. TYPE 1 FORCING *is correct.*

PROOF. If v is a type 1 vertex to a free co-component, by Corollary 4.15, v is a type 1 vertex to all co-components. Now we consider the first case of TYPE 1 FORCING. If there is a fixed co-component $\overline{C}$ such that the adjacent part to v is on the side opposite v, by Lemma 4.16, $\overline{C}$ is the active co-component of v in B and thus the orientations of all free co-components in B are forced (i.e., the remaining free co-components have to be placed such that the adjacent part to v is on the same side as v). Then, all co-components are placed and thus the problem is reduced to only one instance of SBN, we are finished.

So suppose the first case of TYPE 1 FORCING doesn't apply, then every fixed co-component has its adjacent part to v on the same side as v (i.e., they are not the active co-component of v in B). By Lemma 4.16, except possibly the active co-component, if $xy \in E(B)$ and $B^2 = G$, then v is a universal type 1 vertex to every other co-component. This justifies the assertion.

Now we consider the final case of TYPE 1 FORCING. By Lemma 4.16, if $xy \in E(B)$ and $B^2 = G$, then there is at most one co-component $\overline{C}$ such that v is not a type 1 universal vertex to $\overline{C}$. Furthermore, if such a free co-component $\overline{C}$ exists, by Lemma 4.16, $\overline{C}$ must be the active co-component of v and thus the orientation of the remaining free co-components are forced (i.e., the remaining free co-components have to be placed such that the adjacent part to v is on the same side as v). Hence, the problem is again reduced to only one instance of SBN and we are finished. □

COROLLARY 4.18. *Suppose the execution hasn't halted yet after* TYPE 1 FORCING *and let v be a type* 1 *vertex to a free co-component. Then for any fixed co-component $\overline{C}$, the adjacent part of $\overline{C}$ to v is on the same side as v in B. Furthermore, v is a universal type* 1 *vertex to all co-components in G.*

PROOF. Suppose there is a fixed co-component $\overline{C}$ with the adjacent part to v is on the side opposite v, then by the first case of TYPE 1 FORCING, the problem is reduced to one instance of SBN and the execution will halt.

So suppose the execution hasn't halted after the first case, then the assertion in TYPE 1 FORCING ensures that v is a type 1 universal vertex to every fixed co-component. Similarly, if v is not a type 1 universal vertex to a free co-component, then by the final case of TYPE 1 FORCING, the problem is reduced to one instance of SBN and the execution will halt. So we can further assume that v is a universal type 1 vertex to all co-components in G. □

4.4. CHECKPOINT. Now we summarize what we have proved so far. By Remark 4.3, there are two free co-components left. By Corollary 4.14, if $v \in P_{xy}$ is a type 2 vertex to a free co-component, it is a universal type 2 vertex to all free co-components. By Corollary 4.18, if $v \in P_{xy}$ is a type 1 vertex to a free co-component, it is a universal type 1 vertex to all co-components. By Corollary 4.11, v is not a type 0 vertex to any co-component. Henceforth, we refer a vertex of the former case a *type* 2 *vertex* and a vertex of the latter case a *type* 1 *vertex*.

For any type 1 vertex $v \in P_{xy}$, by Lemma 4.16, there is exactly one co-component $\overline{C}$ (the active co-component) such that $\overline{C}$'s adjacent part to v is placed on the side opposite v in B. If we can determine which co-component is the active co-component of v in B, then the orientation of the remaining co-components are forced by Lemma 4.16 (i.e., the remaining co-components have to be placed such that the adjacent part to v is on the same side as v). So suppose there is a type 1 vertex v in P_{xy}. Then we can reduce SBE to at most n instances of SBN by trying every possibility for the active co-component of v. Notice that this observation together with the FINAL PLACEMENT steps already give us an $\mathcal{O}(n \cdot M(n))$ algorithm for SBE, as there are at most n possibilities for the active co-component.

4.4.1. *More Type* 1 *Forcing.* In MORE TYPE 1 FORCING, we search for a free co-component $\overline{C}$ that regardless of $\overline{C}$'s orientation, $\overline{C}$ has to be the active co-component of some type 1 vertex. Suppose we can find such a free co-component $\overline{C}$, then SBE is reduced to at most 2 instances of SBN.

LEMMA 4.19. MORE TYPE 1 FORCING *is correct.*

PROOF. Suppose the first **if** statement is true. By Lemma 4.2, there are only two ways to place $\overline{C}$. Without loss of generality, we assume $\overline{C}$ is placed so that $\overline{C}$'s adjacent part to u is on the side opposite u in B. Recall that by Corollary 4.18, u is a

MORE TYPE 1 FORCING

(1) **if** u and v are type 1 vertices in P_{xy} that are on the same side in B
and there exists a free co-component $\overline{C}$
such that u's and v's adjacent parts on $\overline{C}$ are different
for each orientation of $\overline{C}$
place all free co-components based on Lemma 4.16
if SBN **return** TRUE
return FALSE

(2) **if** u and v are type 1 vertices in P_{xy} that are on different sides in B
and there exists a free co-component $\overline{C}$
such that u's and v's adjacent parts on $\overline{C}$ are the same
for each orientation of $\overline{C}$
place all free co-components based on Lemma 4.16
if SBN **return** TRUE
return FALSE

type 1 universal vertex to every co-component. Since $\overline{C}$ is the active co-component of u in B, by Lemma 4.16, the orientations of the remaining free co-components are forced (the remaining free co-components have to be placed so that the adjacent part to u is on the same side as u) and thus we can apply SBN. The same argument applies if $\overline{C}$ is placed so that $\overline{C}$'s adjacent part to v is on the side opposite v. Therefore, the problem is reduced to at most two instances of SBN. A similar argument applies for the second **if** statement. □

COROLLARY 4.20. *If after the execution of* MORE TYPE 1 FORCING *we still have two free co-components left, then we have the following. For type* 1 *vertices in* P_{xy}*: if they are on the same side, their neighborhood on the free co-components are the same; if they are on different sides, their neighborhoods on the free co-components are disjoint.*

PROOF. Suppose there are two type 1 vertices $u, v \in P_{xy}$ that are on the same side but their neighborhood are not the same. By Corollary 4.18, u and v are universal type 1 vertex to every co-component. Also, by Corollary 4.18, all the fixed co-components have their adjacent part to u and v on the same side as u and v. So, if their neighborhood on the free co-components are not the same, there is a free co-component $\overline{C}$ such that u's and v's adjacent parts to $\overline{C}$ are different. By the first **if** of MORE TYPE 1 FORCING, the problem is reduced to at most two instances of SBN and the execution halts. The other case of the corollary follows similarly. □

4.4.2. *Place Incomplete Co-Components.* We say a co-component $\overline{C}$ is *complete* if there is no edge between the two parts of $\overline{C}$ (i.e., a complete bipartite graph in the complement); otherwise, $\overline{C}$ is an *incomplete* co-component. Now, if there is a type 1 vertex $v \in P_{xy}$, then we will fix the orientations of all free incomplete co-components.

```
PLACE INCOMPLETE CO-COMPONENT
if there is a type 1 vertex v ∈ P_xy
    for any free co-component C̄
        if C̄ is an incomplete co-component in G
            place C̄ such that the adjacent part of C̄ to v
                is on the same side as v
```

LEMMA 4.21. PLACE INCOMPLETE CO-COMPONENT *is correct.*

PROOF. If $v \in P_{xy}$ is a type 1 vertex, by Corollary 4.18, v is a universal type 1 vertex to all free co-components. Now consider an arbitrary incomplete free co-component $\overline{C}$; let w, z be two vertices in different parts in $\overline{C}$ and $wz \in E(G)$. Without loss of generality, we assume w is in the adjacent part of v. By Lemma 4.2, there are only two ways of placing $\overline{C}$. Suppose, by way of contradiction, that $\overline{C}$ is placed such that the adjacent part of v is on the side opposite v. Since $wv \in E(G)$ and $wz \in E(G)$, by Proposition 2.1, $wv \in E(B)$ and $wz \in E(B)$. Therefore, $vz \in E(B^2)$ but this contradicts the assumption that v is a type 1 vertex in G. □

COROLLARY 4.22. *After the execution of* PLACE INCOMPLETE CO-COMPONENT*, if there is a type 1 vertex in* P_{xy}*, then every free co-component is complete.*

4.4.3. *Final Placement I.* If the execution has not halted yet at this point, the unresolved graph has some very special structures. By Remark 4.3, the graph has at least two free co-components. From the discussion of Section 4.4, a vertex in P_{xy} is either a type 1 or a type 2 vertex. Recall that a type 1 vertex is a universal type 1 vertex to all free co-components, and a type 2 vertex is a universal type 2 vertex to all free co-components. We denote the set of type 1 vertices in P_x by P_x^1 and the set of type 2 vertices in P_x by P_x^2; similarly for P_y^1 and P_y^2. By Corollary 4.20, vertices in P_x^1 have the same neighborhood in the free co-components. We denote the adjacent parts of vertices in P_x^1 of the free co-components by $\overline{C}_1^x, \ldots, \overline{C}_k^x$, and similarly, the adjacent parts of vertices in P_y^1 of the free co-components by $\overline{C}_1^y, \ldots, \overline{C}_k^y$. By Corollary 4.20, $\overline{C}_i^x \neq \overline{C}_i^y$ for all i. If $P_x^1 \cup P_y^1 \neq \emptyset$, then by Lemma 4.16, exactly one free co-component $\overline{C}_i$, the *active co-component*, should be placed so that $\overline{C}_i^x$ is in Y and $\overline{C}_i^y$ is in X (in this case, $\overline{C}_i$ is the active co-component of all type 1 vertices in P_{xy} in B). For all other free co-components $\overline{C}_j$ for $i \neq j$, by Lemma 4.16, $\overline{C}_j^x$ should be placed in X and $\overline{C}_j^y$ should be placed in Y. We say such an orientation of all free co-components is *valid*, since only such orientation is allowed by Lemma 4.16. Now we place all free co-components. We have two cases to consider; in FINAL PLACEMENT I, we consider the case when $N_G[x] \cup N_G[y] = V(G)$. In this case, we claim that every valid orientation yield the same squared graph. In FINAL PLACEMENT II, we consider the case when $N_G[x] \cup N_G[y] \subset V(G)$ where we can use an "outside" vertex to determine the orientations of the free co-components in B.

FINAL PLACEMENT I: when $N_G[x] \cup N_G[y] = V(G)$

Case 1: there exists a type 1 vertex in P_{xy}
assert that there is no edge between P_x^1 and P_y
assert that there is no edge between P_y^1 and P_x
place the free co-components by an *arbitrary* valid orientation

Case 2: there is no type 1 vertex in P_{xy}
place the free co-components by an *arbitrary* orientation

return SBN

LEMMA 4.23. *Suppose $B = (X, Y, E)$, $xy \in E(B)$ and $B^2 = G$. If the execution of* SBE *hasn't halted yet and there exists a type 1 vertex in P_{xy}, then there is no edge between P_1^x and P_y and there is no edge between P_1^y and P_x in G.*

PROOF. Suppose, by way of contradiction, there is an edge between $p_1^x \in P_1^x$ and $p^y \in P^y$ in G. Since there exists a type 1 vertex in P_{xy}, as mentioned at the beginning of this subsection, there is exactly one free co-component $\overline{C}$ that has to be placed so that $\overline{C}^x$ is in Y and $\overline{C}^y$ is in X. Let c^y be a vertex in $\overline{C}^y$. Since $p^y c^y \in E(G)$ (regardless of the type of p^y) and $p_1^x p^y \in E(G)$, by Proposition 2.1, $p^y c^y \in E(B)$ and $p_1^x p^y \in E(B)$. Therefore, $c^y p_1^x \in E(B^2)$ but $c^y p_1^x \notin E(G)$ by definition; this contradicts the assumption that $B^2 = G$ and the proof is completed. □

LEMMA 4.24. FINAL PLACEMENT I *is correct.*

PROOF. Recall that by Remark 4.3, there are at least two free co-components $\overline{C}_1$ and $\overline{C}_2$ left. Let B and B' be two bipartite graphs with different (valid) orientations. We claim that $E(B^2) = E(B'^2)$.

Case 1. There exists a type 1 vertex. Without loss of generality, we assume that $\overline{C}_1$, $\overline{C}_2$ are the active co-components of B and B' respectively. Recall that for the active co-component $\overline{C}$, $\overline{C}^x$ is placed in Y and $\overline{C}^y$ is placed in X; while for a remaining co-component $\overline{C}'$, $\overline{C}'^x$ is placed in X and $\overline{C}'^y$ is placed in Y. By Proposition 2.1, if $B^2 = G$, then $E(B) = \{uv \mid u \in X,\ v \in Y \text{ and } uv \in E(G)\}$ and similarly for B'. Recall that for two vertices $u \in X$, $v \in Y$, $uv \in E(B)$ if and only if $uv \in E(B^2)$ since they do not share common neighbor. By our construction of B and B', for $u \in X - \overline{C}_1 - \overline{C}_2$ and $v \in Y - \overline{C}_1 - \overline{C}_2$, $uv \in E(B)$ if and only if $uv \in E(B')$. Therefore, for $u \in X - \overline{C}_1 - \overline{C}_2$ and $v \in Y - \overline{C}_1 - \overline{C}_2$, $uv \in E(B^2)$ if and only if $uv \in E(B'^2)$. Now we consider the case where $u \in X - \overline{C}_1 - \overline{C}_2$ and $v \in X - \overline{C}_1 - \overline{C}_2$. We will show that $uv \in E(B^2)$ and $uv \in E(B'^2)$. Let z be a vertex in C_1^x which is placed in Y in B. Since $N_G[x] \cup N_G[y] = V(G)$, we have $uz \in E(G)$ and $vz \in E(G)$. By Proposition 2.1, $uz \in E(B)$ and $vz \in E(B)$. Therefore, $uv \in E(B^2)$. By a similar argument (by setting z to be a vertex in C_2^x), we have $uv \in E(B'^2)$. Furthermore, the same argument applies for $u \in Y - \overline{C}_1 - \overline{C}_2$ and $v \in Y - \overline{C}_1 - \overline{C}_2$.

Finally, we have to show that the edges with at least one endpoint in $\overline{C}_1 \cup \overline{C}_2$ are the same in B^2 and B'^2. Let $c_1^x, c_2^x, c_1^y, c_2^y$ be an arbitrary vertex in $\overline{C}_1^x, \overline{C}_2^x, \overline{C}_1^y, \overline{C}_2^y$ respectively. We do so by verifying that

(1) $N_{B^2}(c_i^x) = N_{B'^2}(c_i^x) = V(G) - P_1^y - \overline{C}_i^y$ for $i = 1, 2$.

(2) $N_{B^2}(c_i^y) = N_{B'^2}(c_i^y) = V(G) - P_1^x - \overline{C}_i^x$ for $i = 1, 2$.

Recall that in B, $\overline{C}_1^x, \overline{C}_2^y$ are in Y and $\overline{C}_1^y, \overline{C}_2^x$ are in X. First we check the neighbors of c_2^x in B^2. By Corollary 4.22, there is no edge between c_2^x and c_2^y in G and thus in B. Also, there is no edge between c_2^x and vertices in P_1^y in G and thus in B. Since c_2^x is on the side opposite c_2^y and vertices in P_1^y, there is no edge between them in B^2 because they do not share common neighbor in B. On the other hand, c_2^x is adjacent to all vertices in $Y - P_1^y - \overline{C}_2^y$ in B and thus in B^2. Since c_1^x is adjacent to all vertices in $X - \overline{C}_1^y$ in B and c_1^x is adjacent to c_2^x in B, c_2^x is adjacent to all vertices in $X - \overline{C}_1^y$ in B^2. Also, since y is adjacent to all vertices in $\overline{C}_1^y$ in B and y is adjacent to c_2^x in B, c_2^x is adjacent to all vertices in $\overline{C}_1^y$ in B^2. Therefore, $N_{B^2}(c_2^x) = V(G) - P_1^y - \overline{C}_2^y$. The same argument applies and thus we have $N_{B^2}(c_2^y) = V(G) - P_1^x - \overline{C}_2^x$. By a similar argument (by changing the role of $\overline{C}_1$ and $\overline{C}_2$), we have $N_{B'^2}(c_1^x) = V(G) - P_1^y - \overline{C}_1^y$ and $N_{B'^2}(c_1^y) = V(G) - P_1^x - \overline{C}_1^x$.

Now we check the neighbor of c_1^x in B^2. By Corollary 4.22, there is no edge between c_1^x and c_1^y in G and thus in B. Since c_1^x is on the side opposite c_1^y, there is no edge between them in B^2. On the other hand, c_1^x is adjacent to all vertices in $X - \overline{C}_1^x$ in B and thus in B^2. Since c_2^x is adjacent to all vertices in $Y - P_1^y - \overline{C}_2^y$ in B and $c_1^x c_2^x \in E(B)$, c_1^x is adjacent to all vertices in $Y - P_1^y - \overline{C}_2^y$ in B^2. Since x is adjacent to all vertices in $\overline{C}_2^y$ in B and $c_1^x x \in E(B)$, c_1^x is adjacent to all vertices in $\overline{C}_2^y$ in B^2. So, c_1^x is adjacent to $V(G) - P_1^y - \overline{C}_1^y$ in B^2, but not adjacent to $\overline{C}_1^y$ in B^2. Finally, we have to verify that c_1^x is not adjacent to vertices in P_1^y in B^2. By Corollary 4.8, vertices in P_1^y are not adjacent to any trivial co-component in B. By Corollary 4.18, vertices in P_1^y are not adjacent to vertices of fixed co-components in X in G and thus in B. Also, in a valid orientation, vertices in P_1^y are not adjacent to vertices of free co-components in X except C_1^y in B. By Lemma 4.23, vertices in

P_1^y are not adjacent to vertices in P_x in G and thus in B. Therefore, vertices in P_1^y are *only* adjacent to vertices in C_1^y in B. So, c_1^x does not share any common neighbor with vertices in P_1^y in B and thus c_1^x is not adjacent to vertices in P_1^y in B^2. As a result, $N_{B^2}(c_1^x) = V(G) - P_1^y - C_1^y$. The same argument apples and thus we have $N_{B^2}(c_1^y) = V(G) - P_1^x - C_1^x$. By a similar argument (by changing the role of $\overline{C}_1$ and $\overline{C}_2$), we have $N_{B'^2}(c_2^x) = V(G) - P_2^y - \overline{C}_2^y$ and $N_{B'^2}(c_2^y) = V(G) - P_2^x - \overline{C}_2^x$.

Therefore, $E(B^2) = E(B'^2)$. In other words, $B^2 = G$ if and only if $B'^2 = G$. Hence, to construct a bipartite square root B of G, it suffices to consider an arbitrary valid orientation and therefore the problem is reduced to only one instance of SBN.

Case 2. There is no type 1 vertex. In the previous case, the most difficult part is to verify the adjacencies between vertices in P_1^x, P_1^y to the free co-components. In this case, P_1^x, $P_1^y = \emptyset$. By using the arguments in the previous case, one can show that by switching the orientation of one free co-component yields the same squared graph. Hence, to construct a bipartite square root B of G, it suffices to consider an arbitrary orientation and therefore the problem is reduced to only one instance of SBN. □

4.4.4. *Final Placement II.* In FINAL PLACEMENT II, we consider the case when $N_G[x] \cup N_G[y] \subset V(G)$. We will use the adjacencies of a carefully chosen "outside" vertex to decide the orientations of all the free co-components.

LEMMA 4.25. FINAL PLACEMENT II *is correct.*

PROOF. Since $u \notin N_G[x] \cup N_G[y]$, if $B^2 = G$, u is not adjacent to $\{x\} \cup \{y\} \cup C_{xy}$ in B. Since $u \in N_G[C_{xy}] - N_G[x] - N_G[y]$, if $B^2 = G$, u must be adjacent to a vertex $v \in P_{xy}$ in B. Notice that u must exist, otherwise B is disconnected. We will use the adjacencies of u to the free co-components in G to decide the orientations of free co-components in B. Recall that v is either a type 1 or a type 2 vertex in G.

Case 1. There exists $v \in N_B(u) \cap P_{xy}$ that is a type 2 vertex in G. By Corollary 4.14, v is a universal type 2 vertex to every free co-component in G. By Lemma 4.2, every co-component has to be placed so that exactly one part is on the side opposite v. By Proposition 2.1, if $B^2 = G$, v is universally adjacent to exactly one part of each free co-component in B. Hence, u is universally adjacent to exactly one part of each free co-component in $B^2 = G$. Those parts are on the

```
FINAL PLACEMENT II:      when N_G[x] ∪ N_G[y] ⊂ V(G)
    pick a vertex u ∈ N_G[C_xy] − N_G[x] − N_G[y]
    Case 1: u is universally adjacent to exactly one part
              of each free co-component in G
            for each orientation of the free co-components such that
                the adjacent parts of u are on the same side
                    if SBN return TRUE
            return FALSE
    Case 2: u is adjacent to exactly one part of
              exactly one free co-component C̄ in G
            find the valid orientation with C̄ active
            if SBN return TRUE
            else return FALSE
    Otherwise: return NO
```

side opposite v in B. Since there are two possible positions for v (in X or in Y), we try both possibilities. Once we fix all the free co-components, both $N_B(x)$ and $N_B(y)$ are determined; so the problem is reduced to at most 2 instances of SBN.

Case 2. All vertices in $N_B(u) \cap P_{xy}$ are type 1 vertices in G. By Corollary 4.18, v is a universal type 1 vertex to every co-component in G. By Lemma 4.16, exactly one co-component $\overline{C}$ (the active co-component) has to be placed so that $\overline{C}$'s adjacent part is on the side opposite v in B. By Proposition 2.1, if $B^2 = G$, then v is universally adjacent to one part of the active co-component in B but not adjacent to vertices in any other co-component in B. Hence, u is universally adjacent to one part of the active co-component in B^2 but not adjacent to vertices in any other co-component in B^2. This implies that the co-component $\overline{C}$ that u is adjacent to in G is the active co-component in B. By Lemma 4.16, this forces the orientation of all other free co-components; so the problem is reduced to only 1 instance of SBN. □

THEOREM 4.26. SQUARE OF BIPARTITE GRAPH *can be solved in* $\mathcal{O}(\Delta(G) \cdot M(n))$ *time.*

PROOF. After the execution of the FINAL PLACEMENT, any instance of SBE is reduced to at most two instances of SBN. Notice that except the matrix multiplication operation, all the steps can be done in $\mathcal{O}(n^2)$ time. So, SBE can be solved in $\mathcal{O}(M(n))$ time. Therefore, by the argument at the end of Section 2, SQUARE OF BIPARTITE GRAPH can be solved in $\mathcal{O}(\Delta(G) \cdot M(n))$ time. □

5. *Counting and Generating Bipartite Square Roots*

It is natural to ask how many different bipartite square roots a graph can have. In fact, by looking at the SBE algorithm carefully, the only flexibility in the algorithm of placing vertices is in PLACING TRIVIAL CO-COMPONENT when $P_{xy} = \emptyset$ and FINAL PLACEMENT I when $N_G[x] \cup N_G[y] = V(G)$. In the former case, a trivial co-component can be placed in either X or Y; in the latter case, depending on the existence of type 1 vertices, either an arbitrary valid orientation or an arbitrary orientation is considered. As shown before, in either case, two arbitrary placements will have the same squared graph. Hence, when we are just concerned with the existence of a bipartite square root, it suffices to test for an arbitrary placement (i.e., a representative). When we are concerned with the number of different bipartite square roots of G, if the representative is checked to be a bipartite square root of G, then we have to count the number of arbitrary placements, denoted by n_p. We denote the number of trivial co-components by t and the number of free co-components by f. Fortunately, it is easy to count the number of arbitrary placements, summarized as follows:

Case 1. $P_{xy} = \emptyset$
$n_p = 2^{t+f}$

Case 2. $P_{xy} \neq \emptyset$, $V(G) = N_G[x] \cup N_G[y]$ and there is no type 1 vertex
$n_p = 2^f$

Case 3. $P_{xy} \neq \emptyset$, $V(G) = N_G[x] \cup N_G[y]$ and there is a type 1 vertex
$n_p = f$

To see this, for the first case, since $P_{xy} = \emptyset$, no co-components will be fixed before FINAL PLACEMENT I. Hence, every co-component is free to be placed independently, so there are 2^{t+f} possibilities. For the second case, since $P_{xy} \neq \emptyset$, there is no flexibility for the position of trivial co-components. In FINAL PLACEMENT I, every free co-component can be placed arbitrarily and independently, so there are 2^f possibilities. For the last case, since there is a type 1 vertex, we just consider valid orientations and there are exactly f possibilities. In all other cases of SBE, every step is forced. Notice that when the problem is reduced to SBN, the solution is unique by Lemma 3.2.

THEOREM 5.1. *Given G, the number of different bipartite roots $r(G)$ of G can be computed in $\mathcal{O}(\Delta(G) \cdot M(n))$.*

PROOF. As we discussed before, given an instance of SBE, we can determine the number of different solutions. Given an instance of SB, there are at most $\Delta(G)$ instances of SBE. Notice that if we sum the number of solutions of each of the $\Delta(G)$ instances of SBE, we may run into the problem of over-counting.

To overcome the problem of over-counting, we do the following. First, we pick a vertex x with maximum degree in G; we know that it must be a maximal vertex in any bipartite square root B of G. Then, we sort the vertices in $N_G(x)$ by nonincreasing degree. Suppose the resulting ordering is $\{y_1, \ldots, y_k\}$; we reduce SB to k instances of SBE by following the order of the sorted vertices. Consider an instance of SBE of x and y_i; we add additional constraints that $\{y_1, \ldots, y_{i-1}\}$ must be on the same side as x. Notice that by adding these additional constraints, they will not affect the execution of the algorithm. In fact, these just help the algorithm to narrow down the possible choices (i.e., to fix the free co-components).

Now we claim that this algorithm counts the number of different bipartite roots of a graph correctly. Obviously, by adding the additional constraints, we avoid the problem of over-counting. We just have to show that we do not exclude some possible solutions and this completes the proof. First, it is clear that, for all j, we count all solutions with xy_j as a maximal edge and $\{y_1, \ldots, y_{j-1}\}$ in X. The only possible solutions that we may exclude are where xy_j is a maximal edge in the solution but some y_i is also in Y where $i < j$. Consider the smallest such i in the solution; we argue that when we count the solutions of SBE with xy_i as a maximal edge, those solutions are included. The crucial observation is that the only place we use the maximality of y_i in the algorithm is in PLACE TRIVIAL CO-COMPONENTS. In PLACE TRIVIAL CO-COMPONENTS, we exclude a trivial co-component $\overline{c}$ with $N_G(y_i) \subset N_G(\overline{c})$ to be placed in Y by maximality of y_i. Besides that, we do not use the fact that y_i is a maximal vertex in the bipartite square roots. Since $i < j$, $N_G(y_i) \not\subset N_G(y_j)$. Hence, the solutions with y_i and y_j on the same side are included (intuitively speaking, the placement of y_j will not affect the maximality of y_i in the instance of SBE with xy_i as a maximal edge). As a result, this algorithm counts correctly.

Finally, notice that the additional constraints do not increase the complexity of the algorithm. Also, the additional counting step can be performed in linear-time (by just counting the number of trivial co-components and free co-components left); the theorem follows. □

THEOREM 5.2. *Given G, all different bipartite roots of G can be generated in $\mathcal{O}(\max\{\Delta(G) \cdot M(n),\ r(G)\})$.*

In those cases that we may have many different solutions to SBE, we observe that the graph is of small diameter. The following is a sufficient condition for a graph to have a bounded number of different bipartite square roots.

THEOREM 5.3. *Given G, let x be a vertex of maximum degree; if $V(G) \neq N_G[x] \cup N_G[y]$ for any $y \in N_G(x)$, then G has at most $2\Delta(G)$ different bipartite square roots.*

PROOF. In any bipartite square root B of G, x is a maximal vertex in B and there is a vertex $y \in N_G(x)$ such that $y \in N_B(x)$ and y is a maximal vertex in B. Therefore, there are at most $\Delta(G)$ instances of SBE. Since $N_G[x] \cup N_G[y] \neq V(G)$ for all $y \in N_G(x)$, there is no flexibility in the algorithm for SBE and there are at most 2 different solutions for each instance of SBE. Hence, there are at most $2\Delta(G)$ different bipartite square roots. □

6. *Squares of Trees*

Clearly a tree is a bipartite graph. We will use our tools developed for bipartite graphs to give new proofs of some existing results for trees. In particular, we will give a new and much simpler linear-time algorithm to recognize squares of trees [Lin and Skiena 1995], and, as a consequence, a new proof that tree square roots of a graph are unique up to isomorphism, when they exist [Ross and Harary 1960].

6.1. SIMPLE LINEAR-TIME ALGORITHM FOR RECOGNIZING SQUARES OF TREES. First, we show that we can test if $T^2 = G$ in linear time. Notice that this is also shown in Lin and Skiena [1995], but here we give our own proof.

LEMMA 6.1. *Given G and T, testing if $T^2 = G$ can be done in $\mathcal{O}(m)$ where m is the number of edges in G.*

PROOF. Given T, find an arbitrary leaf v of T. Let u be v's parent in T. It is easy to see that $N_{T^2}[v] = N_T[u]$. Therefore, if $T^2 = G$, then $N_T[u]$ must be equal to $N_G[v]$. If $N_T[u] \neq N_G[v]$, we just return "No". Otherwise, we replace T, G by $T - v$, $G - v$ respectively and repeat the process. If only one vertex is remained in T, G, it implies that $N_{T^2}[w] = N_G[w]$ for all $w \in V(G)$. Therefore, $T^2 = G$ and we return "YES" in this case. In each iteration, we remove a vertex v and it takes at most $\mathcal{O}(deg_G(v))$ time. The total time complexity is $\sum \mathcal{O}(deg_G(v)) = \mathcal{O}(m)$. □

We will reduce SQUARE OF TREE to one instance of SBN. By doing so, we first show that if $G = T^2$, then a maximal clique S in G corresponds to $N_T[v]$ for a vertex $v \in S$. In such a case, we call v the *center* of S in T.

LEMMA 6.2. *Suppose $T^2 = G$; if $S \subseteq V(G)$ induces a maximal clique in G, then $S = N_T[v]$ for a vertex $v \in S$.*

PROOF. Clearly, if $V(G) \geq 3$ and $G = T^2$, then $|S| \geq 3$. Since T is a tree, there are two vertices $u, w \in S$ such that $uw \notin E(T)$. Since $uw \in E(G)$ and $G = T^2$, u and w share a common neighbor v in T. First, we claim that v is adjacent to all vertices in $S - v$ in T. Suppose, by way of contradiction, that v is not adjacent to a vertex $z \in S$ in T. Since S induces a maximal clique in G, there is a path P_{uz} in T from u to z of length at most 2 such that $v \notin P_{uz}$. By the same argument, there is a path P_{wz} in T from w to z of length at most 2 such that $v \notin P_{wz}$. Hence, there

are two vertex disjoint paths from u to w in T (uvw and $P_{uz}P_{zw}$); this contradicts the assumption that T is a tree. Therefore, $S \subseteq N_T[v]$. By the maximality of S, $S = N_T[v]$. □

Finding a maximal clique in G can easily be done in linear time by a greedy method. Given a maximal clique S in G, by Lemma 6.2, if we can deduce the center of S, then the problem is reduced to an instance of SBN.

LEMMA 6.3. *Given a maximal clique S in G, if $v_1, v_2 \in S$ share a common neighbor w in $G - S$, then either $N_T[v_1] = S$ or $N_T[v_2] = S$. In other words, either v_1 or v_2 is the center of S.*

PROOF. By Lemma 6.2, $S = N_T[v]$ for a vertex $v \in S$. Suppose, by way of contradiction, that neither v_1 nor v_2 is the center of S. Let the common neighbor of v_1 and v_2 in $G - S$ be w. Since $G = T^2$, there is a path P_{v_1w} from v_1 to w in T of length at most 2 and similarly a path P_{v_2w} from v_2 to w in T of length at most 2. Since $w \notin N_T[v]$, $v \notin P_{v_1w}$ and $v \notin P_{v_2w}$. Therefore, there are two vertex disjoint paths from v_1 to v_2 in T (v_1vv_2 and $P_{v_1w}P_{wv_2}$); a contradiction. □

A natural approach to finding a tree root of a graph is to identify the leaves and their parents, and repeat the process recursively. In fact, it is the approach used in Lin and Skiena [1995] and Kearney and Corneil [1998]. Notice that Lin and Skiena [1995] gave a linear time algorithm to find a tree square root of a graph, here we use a different approach to give a simpler linear time algorithm.

THEOREM 6.4 (SEE ALSO LIN AND SKIENA [1995]). SQUARE OF TREE *can be solved in linear time.*

PROOF. First of all, we find an arbitrary maximal clique S in G. By Lemma 6.2, S corresponds to $N_T[v]$ for a $v \in S$.

Case 1. $S = V(G)$. In this case, G is a complete graph and any complete star is a tree square root of G.

Case 2. $S \subset V(G)$. By Lemma 6.3, if two vertices v_1, v_2 in S share a common neighbor in $G - S$ in G, then one of them is the center. It is easy to see, if T is connected, there is at least one such pair of vertices.

Case 2a. There are at least two distinct pairs of vertices. We pick two arbitrary distinct pairs. By Lemma 6.3, if $G = T^2$, there is exactly one vertex v (the center) that appears in more than one pair and thus $N_T[v] = S$. So, in this case, the problem is reduced to an instance of SBN.

Case 2b. There is only one distinct pair of vertices. Suppose the center of S in T is v, it is easy to see that all vertices in $S - v$ are leaves in T except exactly one internal vertex u. So $N_{T^2}(v) \subseteq N_{T^2}(u)$. Let v_1, v_2 be the only pair of vertices. Since $G = T^2$, if $N_G(v_1) \subset N_G(v_2)$, then v_1 must be the center and thus the problem is reduced to an instance of SBN. The only case left is when $N_G(v_1) = N_G(v_2)$. In this case, it implies the tree is of diameter 3 (and thus a double star, see Figure 3), and thus v_1 and v_2 are indistinguishable (i.e., there are two different but isomorphic tree square roots).

Now we show that the algorithm can be implemented in linear time. As we discussed before, a maximal clique S in G can be found in linear time. To find a

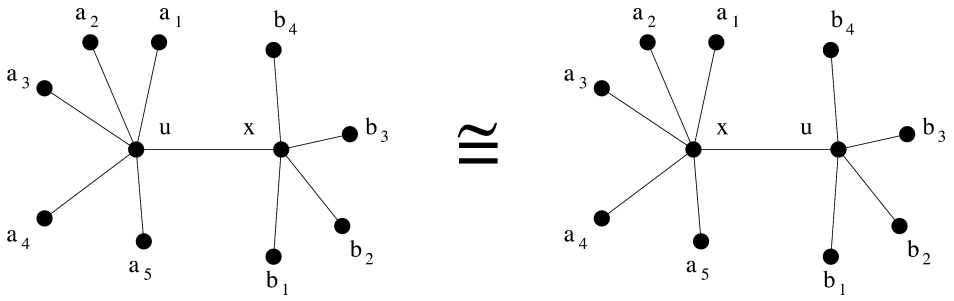

FIG. 3. Two isomorphic but not equivalent double stars.

pair of vertices in S that share a common neighbor in $G - S$, it suffices to check the neighborhood in S for every vertex in $G - S$; every edge is visited at most once. Once we find two distinct pairs, we can reduce the problem to an instance of SBN. So at any time of the algorithm, we just have to store one such pair of vertices. Notice that in any case, the problem is reduced to at most one instance of SBN. As mentioned in the proof of Lemma 3.2, the unique candidate can be constructed in linear time. Then we test if the unique candidate is a tree. If it is, by Lemma 6.1, the solution can be verified in linear time and we are done. □

Ross and Harary [1960] showed that tree square roots of a graph, when they exist, are unique up to isomorphism. Their proof is based on a characterization of tree squares. We now give an algorithmic proof based on our algorithm.

THEOREM 6.5 (SEE ALSO ROSS AND HARARY [1960]). *Tree square roots of a graph, when they exist, are unique up to isomorphism.*

PROOF. From the proof of Theorem 6.4, there are only two cases where we can not pin down exactly the center of the maximal clique. The first case is when the tree square root is a star while the second case is when the tree square root is a double star. In both cases, the tree square roots of G are isomorphic. Note that if we can pin down the center of the maximal clique, then by Lemma 3.2, the solution is unique. □

COROLLARY 6.6. *If $G = T^2$ for some T of diameter greater than* 3*, then T is the unique tree square root of G.*

7. *Cubes of Bipartite Graphs*

Since SQUARE OF BIPARTITE GRAPH is polynomial time solvable, it is natural to ask if we can find a bipartite k-th root of a graph in polynomial time for $k \geq 3$. We observe that Proposition 2.1 does not hold for $k \geq 3$. In fact, we will show in this section that it is NP-complete to determine if a given graph G is the cube of a bipartite graph.

PROBLEM. CUBE OF BIPARTITE GRAPH
INSTANCE. A graph $G = (V, E)$.
QUESTION. Does there exist a *bipartite graph* B such that $G = B^3$?

In our reduction, we use SET SPLITTING as formulated in Garey and Johnson [1979].

PROBLEM.	[SP4] SET SPLITTING
INSTANCE.	Collection C of finite sets of elements from S, positive integer $K \leq \|C\|$.
QUESTION.	Is there a partition of S into two subsets S_1 and S_2 such that no sub set in C is entirely contained in either S_1 or S_2?
NOTE.	It is also known as HYPERGRAPH 2-COLORABILITY.

7.1. TAIL STRUCTURE. In Motwani and Sudan [1994], the tail structure of a vertex v was introduced to ensure v has the same neighborhood in any square root H of G. It enables one to pin down exactly the neighborhood of v in any square root H of G. We generalize the tail structure of a vertex v such that v has the same neighborhood in any kth root H of G. This enables us to pin down exactly the neighborhood of v in any kth root H of G. Notice that these results apply to general graphs G and H.

LEMMA 7.1. *Let G be a connected graph with $\{v_1, \ldots, v_{k+1}\} \subset V(G)$ where*

—$N_G(v_1) = \{v_2, \ldots, v_{k+1}\}$
—$N_G(v_i) \subset N_G(v_{i+1})$ *for all* $1 \leq i \leq k$

Then in any kth *root H of G,*

—$N_H(v_1) = \{v_2\}$
—$N_H(v_i) = \{v_{i-1}, v_{i+1}\}$ *for all* $2 \leq i \leq k$
—$N_H(v_{k+1}) - v_k = N_G(v_2) - \{v_1, \ldots, v_{k+1}\}$.

PROOF. Since $N_{H^k}(v_1) = N_H^1(v_1) \cup \cdots \cup N_H^k(v_1)$, if $H^k = G$, then $N_{H^k}(v_1) = N_G(v_1)$ and thus $N_H^1(v_1) \cup \cdots \cup N_H^k(v_1) = \{v_2, \ldots, v_{k+1}\}$. Since v_1 is not a universal vertex in G, there is a vertex u such that $d_H(u, v_1) > k$ and thus $N_H^i(v_1) \neq \emptyset$ for $1 \leq i \leq k$. Otherwise, there is no path between u and v_1 in H and thus there is no path between u and v_1 in G which contradicts the assumption that G is connected. Since $N_{H^k}(v_1)$ is the disjoint union of k sets $(N_H^1(v_1), \ldots, N_H^k(v_1))$ and $N_{H^k}(v_1)$ contains k vertices only, in order to satisfy the constraint that $N_H^i(v_1) \neq \emptyset$ for $1 \leq i \leq k$, the only possibility is $|N_H^i(v_1)| = 1$ for $1 \leq i \leq k$. Therefore, it forces the path structure in H (i.e., the neighbors of $N_H^i(v_1)$ in H are $N_H^{i-1}(v_1)$ and $N_H^{i+1}(v_1)$ for $i > 1$). As it is a path structure, if $u \in N_H^i(v_1)$ and $w \in N_H^{i+1}(v_1)$, then $N_{H^k}(u) \subseteq N_{H^k}(w)$. Since $N_G(v_i) \subset N_G(v_{i+1})$ and $|N_H^i(v_1)| = 1$ for $1 \leq i \leq k$, if $H^k = G$, we must have $N_H^i(v_1) = \{v_{i+1}\}$. Therefore, $N_H(v_1) = \{v_2\}$ and $N_H(v_i) = \{v_{i-1}, v_{i+1}\}$ for all $2 \leq i \leq k$. The first two properties of the lemma are satisfied. The final property is just a consequence of the first two properties. With the first two properties satisfied, $N_{H^{k-1}}(v_2) = \{v_1, \ldots, v_{k+1}\}$. With one more step, $N_{H^k}(v_2) = \{v_1, \ldots, v_{k+1}\} \cup N_H(v_{k+1})$. So if $H^k = G$, then $N_G(v_2) - \{v_1, \ldots, v_{k+1}\} = N_H(v_{k+1}) - v_k$. □

In order words, v_2 (as seen in G) exactly pins down the neighborhood of v_{k+1} in any kth root of G. We refer to the vertices $\{v_1, \ldots, v_k\}$ as the "tail vertices" of v_{k+1} (see Figure 4).

7.2. THE REDUCTION. The rest of this section shows that CUBE OF BIPARTITE GRAPH is NP-hard by reducing SET SPLITTING to it. It is clear that CUBE OF BIPARTITE GRAPH is in NP, since guessing the cube root B, verifying that B is a bipartite

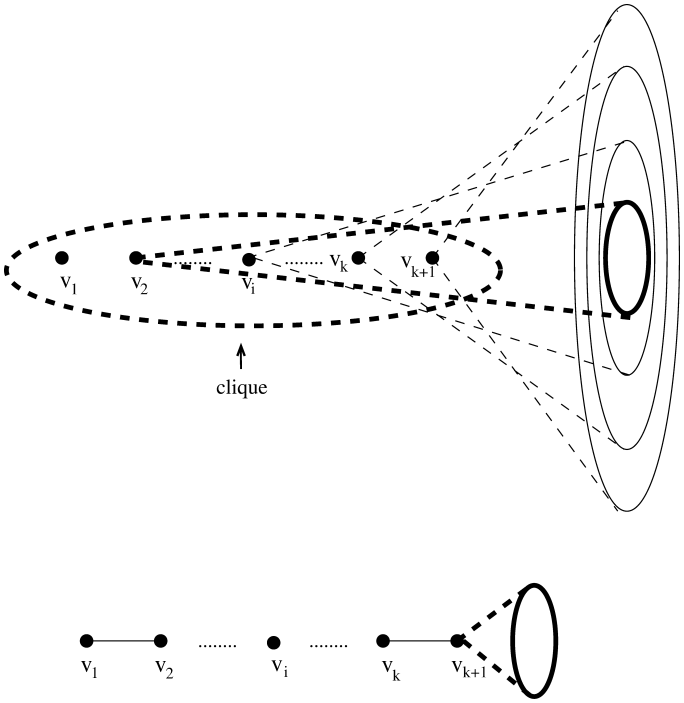

FIG. 4. Tail in $G = H^k$ and H.

graph and $G = B^3$ can easily be done in polynomial stime. Thus, we will conclude that CUBE OF BIPARTITE GRAPH is NP-complete.

Given an instance of SET SPLITTING, we construct an instance of CUBE OF BIPARTITE GRAPH. Let $C = \{c_1, \ldots, c_m\}$ denote the set of subsets where c_j is the set of elements in subset j. And let $S = \{u_1, \ldots, u_n\}$ be the ground set. The graph G is constructed as follows (note that we will be using Lemma 7.1):

Vertices of G

—*Element vertices:* U_i: $1 \leq i \leq n$ for each element u_i.
—*Subset vertices:* C_j for each subset $c_j \in C$ and tail vertices C_j^1, C_j^2, C_j^3 for each c_j.
—*Partition vertices:* S_1 and S_2.
—*Connection vertex:* X.

Edges of G

—*Edges of tail vertices of subset vertices:* $\forall c_j \in C$,
$C_j^3 \leftrightarrow C_j^2, C_j^3 \leftrightarrow C_j^1, C_j^3 \leftrightarrow C_j$
$C_j^2 \leftrightarrow C_j^1, C_j^2 \leftrightarrow C_j, C_j^2 \leftrightarrow U_i$ for all $u_i \in c_j$,
$C_j^1 \leftrightarrow C_j, C_j^1 \leftrightarrow C_i$ iff $c_j \cap c_i \neq \emptyset, C_j^1 \leftrightarrow U_i$ iff $u_i \in c_j, C_j^1 \leftrightarrow S_1, C_j^1 \leftrightarrow S_2, C_j^1 \leftrightarrow X$.
—*Edges of subset vertices:* $\forall c_j \in C$,
$C_j \leftrightarrow S_1, C_j \leftrightarrow S_2, C_j \leftrightarrow X, C_j \leftrightarrow U_i$ for all $i, C_j \leftrightarrow C_i$ iff $c_j \cap c_i \neq \emptyset$.

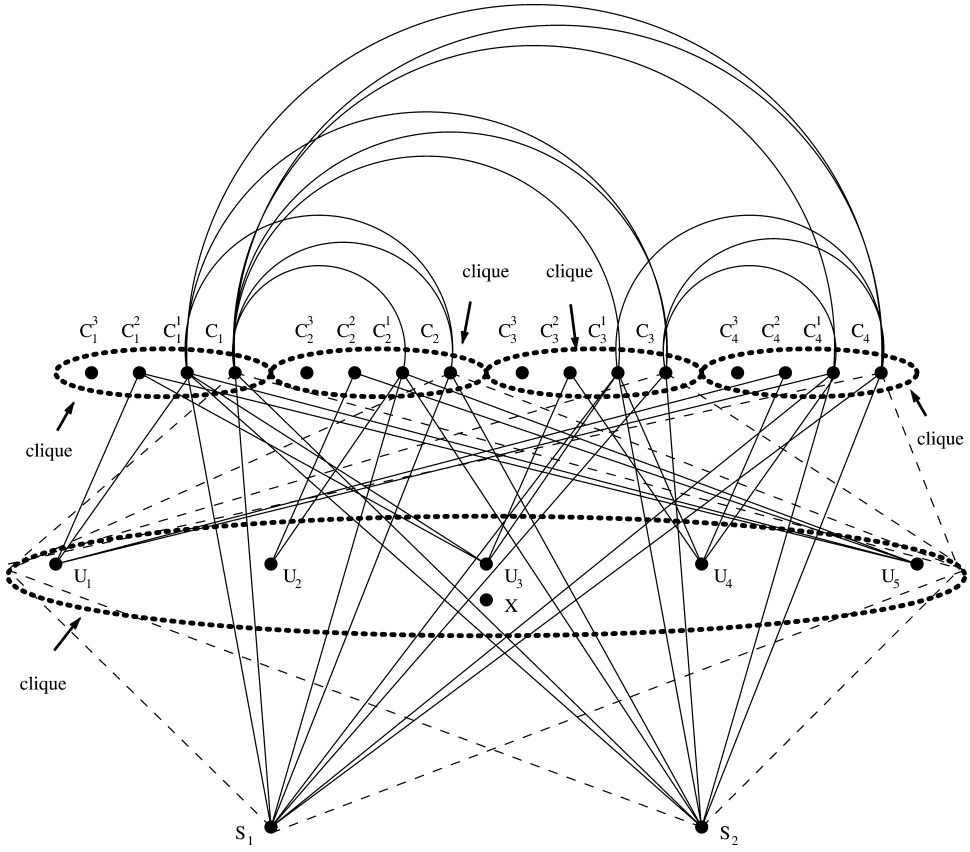

FIG. 5. An example of G.

—*Edges of element vertices:* $\forall u_j \in S$,
$U_j \leftrightarrow U_i$ for all i, $U_j \leftrightarrow S_1$, $U_j \leftrightarrow S_2$, $U_j \leftrightarrow X$.

—*Edges of partition vertices:* $S_1 \leftrightarrow X$, $S_2 \leftrightarrow X$.

For example, given $C = \{c_1, c_2, c_3, c_4\}$, $c_1 = \{u_1, u_2, u_3\}$, $c_2 = \{u_2, u_5\}$, $c_3 = \{u_3, u_4\}$ and $c_4 = \{u_1, u_4\}$ and $S = \{u_1, u_2, u_3, u_4, u_5\}$, we construct G as shown in Figure 5. It is fairly complicated as the cube of a graph introduces many edges. Each ellipse corresponds to a clique and we omit the clique edges to keep the figure simpler. Also in the figure, C_1, C_2, C_3, C_4, S_1 and S_2 have two dotted lines to the central ellipse. This represents that each of them has all the edges to the vertices in the central ellipse.

In this example, $S_1 = \{u_1, u_3, u_5\}$ and $S_2 = \{u_2, u_4\}$ is a possible solution. The corresponding graph B is shown in Figure 6. The reader may verify that $B^3 = G$. Also, a proper 2-coloring of B is given in Figure 6 to show that B is bipartite.

LEMMA 7.2. *If there is a partition of S into two subsets S_1 and S_2 such that no subset in C is entirely contained in either S_1 or S_2, then there exists a bipartite graph B such that $B^3 = G$.*

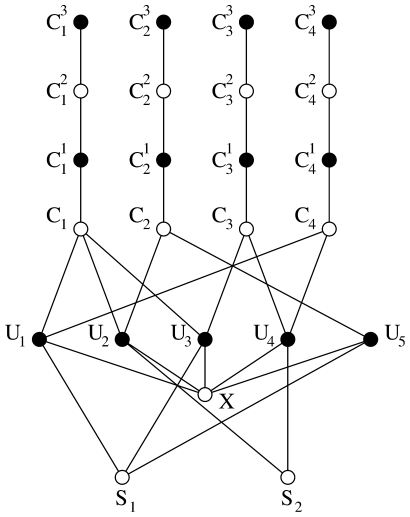

FIG. 6. An example of B.

PROOF. **Edges of B.**

—*Edges of subset vertices and its tail vertices:*
$C_j^3 \leftrightarrow C_j^2$, $C_j^2 \leftrightarrow C_j^1$, $C_j^1 \leftrightarrow C_j$ and $C_j \leftrightarrow U_i$ if and only if $u_i \in c_j$.

—*Edges of partition vertices:*
$S_k \leftrightarrow U_i$ if and only if $u_i \in S_k$.

—*Edges of connection vertex:*
$X \leftrightarrow U_i$ for all i.

If a vertex u has a path of length at most k to v, then we say u *reaches* v in k steps. Now we verify that in general the edge set of B^3 is equal to the edge set of G. We do this by following the order of the presentation of the edge set of G above. It is clear that C_j^3 is only adjacent to C_j^2, C_j^1 and C_j in B^3 for all j.

The vertex C_j^2 reaches C_j in B in two steps. Since $C_j \leftrightarrow U_i$ in B if and only if $u_i \in c_j$, $C_j^2 \leftrightarrow U_i$ in B^3 if and only if $u_i \in c_j$. So $N_{B^3}(C_j^2) = N_G(C_j^2)$ for all j.

The vertex C_j^1 reaches C_j in B in one step. In the second step, it reaches U_i iff $u_i \in c_j$ in B. Since we have a solution to the set splitting, every C_j has a common neighbor to S_1 and a common neighbor to S_2 in B. Furthermore, every U_i is adjacent to X in B and thus C_j^1 reaches S_1, S_2 and X in B in the third step. Also, C_j and C_i share a common neighbor U_k in B iff $c_i \cap c_j \neq \emptyset$. So C_j^1 reaches C_i in B in the third step iff $c_i \cap c_j \neq \emptyset$. We have $N_{B^3}(C_j^1) = N_G(C_j^1)$ for all j.

The vertex C_j reaches U_i in B in one step iff $u_i \in c_j$. Since we have a solution to the set splitting, every C_j has a common neighbor to S_1 and a common neighbor to S_2 in B. Furthermore, every U_i is adjacent to X in B and thus C_j reaches S_1, S_2 and X in B in the second step. Then C_j reaches U_i for all i such that $u_i \notin c_j$ in B in the third step through X. So C_j reaches all U_i in B in three steps. Also, C_j and C_i share a common neighbor U_k in B iff $c_i \cap c_j \neq \emptyset$. So, C_j reaches C_i in two steps if $c_i \cap c_j \neq \emptyset$. On the other hand, if $c_i \cap c_j = \emptyset$, $C_i \not\leftrightarrow C_j$ since $U_h \not\leftrightarrow U_k$ for all h, k. So C_j reaches C_i in three steps iff $c_i \cap c_j \neq \emptyset$. Therefore, $N_{B^3}(C_j) = N_G(C_j)$ for all j.

The vertex U_j is adjacent to X in B and thus it reaches all U_i in B in two steps and then reaches S_1 and S_2 in B within three steps. So, $N_{B^3}(U_j) = N_G(U_j)$ for all j.

S_1 and S_2 reach X in B in two steps. It should be pointed out that $S_1 \not\leftrightarrow S_2$ because it is a partition and thus S_1 and S_2 share no common elements. So, $N_{B^3}(S_j) = N_G(S_j)$ for all j.

We checked that the edge set of B^3 is equal to the edge set of G. Now we show that B is a bipartite graph thereby completing the proof. We do this by finding a 2-coloring of B. S_1, S_2, X, C_j, C_j^2 for all j get color 1. U_i for all i, C_j^1, C_j^3 for all j get color 2. It is a routine matter to check that vertices in the same color class are not adjacent. □

We now show that if G has a cube root H (H may not be a bipartite graph), then there is a partition of S into two subsets S_1 and S_2 such that no subset in C is entirely contained in either S_1 and S_2. First, observing that $\{C_j^3, C_j^2, C_j^1, C_j\}$ satisfies the properties of Lemma 7.1, we have the following consequence:

PROPOSITION 7.3. *If H is a cube root of G, then, in H, C_j is adjacent to U_i if and only if $u_i \in c_j$. Also, in H, C_j^3 is only adjacent to C_j^2, C_j^2 is only adjacent to C_j^1 and C_j^3 and C_j^1 is only adjacent to C_j and C_j^2.*

Now we are ready to prove the reverse direction.

LEMMA 7.4. *If H is a cube root of G, then there is a partition of S into two sets S_1 and S_2 such that no subset in C is entirely contained in either S_1 or S_2.*

PROOF. In H, Proposition 7.3 forces the adjacencies of the tail vertices and subset vertices to its own elements only. So in H, S_1 and S_2 only have neighbors in the element set and X. Since $S_1 \not\leftrightarrow S_2$ in G, they have no common element neighbor in H and so it is a partition. And since S_1 and S_2 are adjacent to all C_j^1, S_1 and S_2 must reach C_j in exactly two steps and thus each of S_1, S_2 must have a common neighbor with C_j in the element set for all j. Therefore, $N_H(S_1) \cap S$ and $N_H(S_2) \cap S$ is the desired partition; this completes the proof. □

Notice that in the above lemma, we did not use the property that H is a bipartite graph. In fact, *any cube root* of G will do. In particular, *any bipartite cube root B* of G will do. We state it formally as follows.

LEMMA 7.5. *If B is a cube root of G and B is bipartite, then there is a partition of S into two subsets S_1 and S_2 such that no subset in C is entirely contained in either S_1 or S_2.*

The following theorem follows immediately from Lemma 7.2 and Lemma 7.5.

THEOREM 7.6. CUBE OF BIPARTITE GRAPH *is NP-complete.*

7.3. CUBES OF GRAPHS.

PROBLEM. CUBE OF GRAPH
INSTANCE. A graph $G = (V, E)$.
QUESTION. Does there exist a graph H such that $G = H^3$?

In Lemma 7.2, we showed that if there is a partition, then we can construct a bipartite graph B such that $B^3 = G$. Since bipartite graph is a special case of general graph, the following is a consequence.

THEOREM 7.7. CUBE OF GRAPH *is NP-complete.*

Although it has been generally expected that CUBE OF GRAPH is NP-complete, Theorem 7.7 is the first result to show that it is indeed NP-complete. Extending the NP-completeness to kth power for any fixed $k > 3$, however, may need a more general reduction. Nonetheless, we strongly believe that the following problems are NP-complete.

PROBLEM. kTH POWER OF GRAPH
INSTANCE. A graph $G = (V, E)$.
QUESTION. Does there exist a graph H such that $G = H^k$ for a fixed $k \geq 2$?

PROBLEM. kTH POWER OF BIPARTITE GRAPH
INSTANCE. A graph $G = (V, E)$.
QUESTION. Does there exist a *bipartite graph* B such that $G = B^k$ for a fixed $k \geq 3$?

8. *Concluding Remarks*

We presented a $\mathcal{O}(\Delta(G) \cdot M(n))$ algorithm to find a bipartite square root and count the number of different bipartite square roots of a graph. The obvious question to ask for this algorithm is: can we reduce SB to a constant number of instances of SBE? If we can do this faster than doing matrix multiplication, then we can obtain an $\mathcal{O}(M(n))$ algorithm for finding a bipartite square root of a graph which matches the complexity of computing the square of a bipartite graph.

Taking a more general perspective (i.e., a tree is a bipartite graph), we give a new and much simpler algorithm to find a tree square root of a graph, if it exists. It is interesting to see if there is a faster algorithm to find a tree kth root of a graph (the current complexity is $\mathcal{O}(n^3)$ by Kearney and Corneil [1998]).

ACKNOWLEDGMENTS. I am very grateful to my supervisor Derek Corneil and two anonymous referees for many helpful suggestions.

REFERENCES

ESCALANTE, F., MONTEJANO, L., AND ROJANO, T. 1974. Characterization of n-path graphs and of graphs having nth root. *J. Combin. Theory B 16*, 282–289.

GAREY, M. R., AND JOHNSON, D. S. 1979. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, Oxford, UK.

GELLER, D. P. 1968. The square root of a digraph. *J. Combin. Theory B 5*, 320–321.

HARARY, F., KARP, R. M., AND TUTTE, W. T. 1967. A criterion for planarity of the square of a graph. *J. Combin. Theory 2*, 395–405.

KEARNEY, P., AND CORNEIL, D. 1998. Tree powers. *J. Algor. 29*, 111–131.

LAU, L. C., AND CORNEIL, D. 2004. Recognizing powers of proper interval, split and chordal graphs. *SIAM J. Disc. Math. 18*, 1, 83–102.

LIN, Y. L., AND SKIENA, S. 1995. Algorithms for square roots of graphs. *SIAM J. Disc. Math. 8*, 1, 99–118.

MOTWANI, R., AND SUDAN, M. 1994. Computing roots of graphs is hard. *Disc. Appl. Math. 54*, 81–88.

MUKHOPADHYAY, A. 1967. The square root of a graph. *J. Combin. Theory 2*, 290–295.

ROSS, I. C., AND HARARY, F. 1960. The square of a tree. *Bell Syst. Tech. J. 39*, 641–647.

WEST, D. 2001. *Introduction to Graph Theory*, Ed. 2, Prentice-Hall, Englewood Cliffs, NJ.

RECEIVED JULY 2004; ACCEPTED APRIL 2005