RECOGNIZING POWERS OF PROPER INTERVAL, SPLIT AND CHORDAL GRAPHS

LAP CHI LAU * AND DEREK G. CORNEIL †

Abstract. In this paper, we study the complexity of recognizing powers of chordal graphs and its subclasses. We present the first polynomial time algorithm to recognize squares of proper interval graphs and give an outline of an algorithm to recognize k-th powers of proper interval graphs for every natural number k. These are the first results of this type for a family of graphs that contains arbitrarily large cliques.

On the other hand, we show the NP-completeness of recognizing squares of chordal graphs, recognizing squares of split graphs and recognizing chordal graphs that are squares of some graph.

Key words. proper interval graphs, split graphs, chordal graphs, graph roots, graph powers, graph algorithms.

AMS subject classifications. 05C12, 05C75, 68R10

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. Note that the terms "power" and "root" are used because of the close relationship with matrix multiplication. Graph roots are associated with problems in distributed computing [26] and computational biology [31, 24] where graph roots are useful in the reconstruction of phylogeny.

For any class of graphs, recognition is a fundamental structural and algorithmic problem; in this paper, we study recognition problems on graph powers. Generally, it is a difficult task to determine whether a given graph G has a k-th root or not. In 1967, Mukhopadhyay [30] characterized general graphs which possess a square root and in the following year Geller [13] solved the problem for general digraphs. In 1974, Escalante, Montejano and Rojano [9] characterized graphs and digraphs with a k-th root. However, all of these characterizations on powers of general graphs are not polynomial in the sense that they do not yield a polynomial time algorithm. The complexity of graph power recognition was unresolved until 1994 when Motwani and Sudan [29] proved the NP-completeness of recognizing squares of graphs. About the same time, Lin and Skiena [25] gave a linear time algorithm to recognize squares of trees. Somewhat surprisingly, until very recently, trees are the only non-trivial family of graphs where the square recognition problem is known to have a polynomial time algorithm (for classes of graphs with diameter at most 2, such as the class of cographs, the square recognition problem is trivial since the square is always a clique). In [22], Lau presented a polynomial time algorithm to recognize squares of bipartite graphs (it is worth noting that Motwani and Sudan [29] believed that this problem would be NP-complete). In this paper, we give a polynomial time algorithm to recognize squares of proper interval graphs. This is the first class of graphs that does not contain trees and furthermore allows graphs with arbitrarily large cliques (the algorithms for trees and bipartite graphs are very dependent on the existence of

*Department of Computer Science, University of Toronto, 10 King's College Road, M5S 3G4, Canada (chi@cs.toronto.edu).

†Department of Computer Science, University of Toronto, 10 King's College Road, M5S 3G4, Canada (dgc@cs.toronto.edu).

no cliques of size > 2). An obvious extension of the square recognition problem is that of k-th power recognition. For $k > 2$ it was solved for trees in polynomial time [18]; for bipartite graphs it is NP-complete [22]. In this paper we show that for proper interval graphs it is in P, for every fixed k. In [16], Harary and McKee introduced the closed-neighborhood intersection multigraph as a useful multigraph version of the square of a graph. They characterized those multigraphs which are squares of chordal graphs and gave an algorithm to go from the squared chordal graph back to its unique square root. In this paper, we show the NP-completeness of recognizing squares of chordal graphs. Also, we prove the NP-completeness of recognizing squares of split graphs and recognizing chordal graphs that are squares of some graph. Flotow [11] studied a related problem to graph powers recognition; he gave sufficient conditions for a graph whose power is a chordal graph (or an interval graph).

1.1. Overview of this paper. In this paper, we study the complexity of recognizing powers of chordal graphs and its subclasses. In section 2, we present the first polynomial time algorithm to recognize squares of proper interval graphs. And we sketch an outline of a polynomial time algorithm to recognize k-th powers of proper interval graphs for every natural number k. Our approach is based on a non-trivial use of dynamic programming. In section 3, we prove the NP-completeness of recognizing squares of chordal graphs, recognizing squares of split graphs, and recognizing chordal graphs that are squares of some graph. Note that split graphs and bipartite graphs have a similar partitioning structure but squares of bipartite graphs can be recognized in polynomial time [22]. Before presenting our results we survey some important results concerning graph powers and present our terminology.

1.2. Related work. The literature is rich with results on graph roots and powers. Given a graph G with property P, does G^k have property P? Substantial work has been done on closure properties of powers of special classes of graphs, such as chordal graphs [1, 8], interval graphs [33], co-comparability graphs [6], strongly chordal graphs [27, 34, 2], circular arc graphs [34], and AT-free graphs [33, 3]. Given a graph G, what can be said about the properties of G^k? Since the number of edges increases with the index of the power of a graph, it is natural to expect that sufficiently large powers do possess some Hamiltonian type properties. For instance, Fleischner [10] proved that the square of every 2-connected graph is Hamiltonian and Sekanina [36] proved that the cube of every non-trivial connected graph is Hamiltonian connected. Besides the mathematical property questions on graph powers, the following is an obvious question to ask from an algorithmic point of view. Given a graph G, can we solve some optimization problems on G^k efficiently? Many optimization problems remain difficult in the case of powers of graphs [25]. On the other hand, a Hamiltonian cycle in the square of a 2-connected graph can be found in polynomial time [21] and the chromatic number of the square of a planar graph can be approximated within a constant factor in polynomial time [28]. Also, Ramachandran [32] proved, without using computers, that if G is a planar graph with a square root or a cube root, then G is 4 colorable.

1.3. Basic terminology. Our basic notation and terminology reference is [38]. We denote a *graph* G with *vertex set* $V(G)$ and *edge set* $E(G)$ by $G = (V, E)$ where n and m denote $|V|$ and $|E|$ respectively. A *loop* is an edge whose endpoints are equal. *Multiple edges* are edges having the same pair of endpoints. A *simple graph* is a graph having no loops or multiple edges. 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. A *pendant vertex* is a vertex with degree one. An *isolated vertex* is a vertex with degree zero. The *open neighborhood* of v, written $N_G(v)$ or $N(v)$, is the set of vertices adjacent to v. The *closed neighborhood* of v, written $N_G[v]$, 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 a u, v-path. The *k-th neighborhood* of v, written $N_G^k(v)$, is the set of vertices with distance k to v. A graph G is *connected* if each pair of vertices in G is connected by a path; otherwise, G is *disconnected*. 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.

A graph G is *chordal* if G does not have an induced cycle of length at least 4. A vertex v is *simplicial* if $G[N(v)]$ is a clique. Let $\sigma = [v_1, v_2, \ldots, v_n]$ be an ordering of the vertices in a graph G. We say that σ is a *simplicial elimination ordering* if each v_i is a simplicial vertex of $G[\{v_i, \ldots, v_n\}]$. It is well known that a graph is chordal if and only if it has a simplicial elimination ordering. A graph is *weakly chordal* if G and $\overline{G}$ contain no induced cycle of length at least 5. Let $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ be a finite collection of intervals of the real line and let $G_{\mathcal{I}}$ be its intersection graph. G is an *interval graph* if G is the intersection graph $G_{\mathcal{I}}$ of an interval model $\mathcal{I}$. G is a *proper interval graph* if G is an interval graph with an interval model where no two intervals $I_x, I_y \in \mathcal{I}$ properly contain each other. Furthermore, G is a proper interval graph if and only if it is a *unit interval graph*, namely an interval graph where all intervals are of the same length. Both interval graphs and proper interval graphs [5, 7] can be recognized in linear time. Furthermore, a fully dynamic algorithm for recognizing and representing proper interval graphs is available [17]. A graph is a *split graph* if there is a partition of its vertex set into a clique and a stable set. Clearly split graphs are chordal.

2. Squares of Proper Interval Graphs. In this section, we will present the first polynomial time algorithm to solve SQUARE OF PROPER INTERVAL GRAPH.

PROBLEM SQUARE OF PROPER INTERVAL GRAPH
INSTANCE A graph $G = (V, E)$.
QUESTION Does there exist a *proper interval graph* H such that $H^2 = G$?

The algorithm is based on a dynamic programming approach and it is conceptually simple. The outline of this section is as follows. First we develop some special structural properties of the square of a proper interval graph. This gives us insight into a polynomial time recognition algorithm. To facilitate our discussion, we then introduce the notation for the description of our algorithm. Then, we prove some lemmas and a decomposition theorem which is the core of our algorithm. And then we present the algorithm formally, prove its correctness and analyze the complexity. Finally, we give an outline to extend the algorithm to recognize k-th powers of proper interval graphs for every natural number k.

PROBLEM k-TH POWER OF PROPER INTERVAL GRAPH
INSTANCE A graph $G = (V, E)$.
QUESTION Does there exist a *proper interval graph* H such that $H^k = G$?

2.1. Preliminaries. Let G be a proper interval graph and $V(G) = \{g_1, \ldots, g_n\}$. Let $I_G(g_i)$ be the corresponding interval of g_i. We denote the *left endpoint* of $I_G(g_i)$ by $left_G(g_i)$ and the *right endpoint* of $I_G(g_i)$ by $right_G(g_i)$. In the remainder of this section, we will assume that $left_G(g_1) < left_G(g_2) < \ldots < left_G(g_n)$. Since G is a proper interval graph, $left_G(g_i) < left_G(g_j)$ implies $right_G(g_i) < right_G(g_j)$. We call such an ordering a *total vertex ordering* in a proper interval graph.

A set S of vertices (intervals) is *consecutive* if $S = \{g_i, g_{i+1}, g_{i+2}, \ldots, g_j\}$ for some $1 \leq i \leq j \leq n$ and we define $left_G(S) = left_G(g_i)$ and $right_G(S) = right_G(g_j)$. Given two non-empty sets of consecutive vertices S_1 and S_2, we say $S_1 < S_2$ if and only if S_1 and S_2 are disjoint and $left_G(S_1) < left_G(S_2)$. For technical reasons, we say $S_1 < S_2$ when $S_1 = \emptyset$ or $S_2 = \emptyset$.

A graph class $\mathcal{C}$ is *closed under powers* if for every $G \in \mathcal{C}$ and every k, $G^k \in \mathcal{C}$. $\mathcal{C}$ is *strongly closed under powers* if $G^k \in \mathcal{C}$ for some k implies $G^{k+1} \in \mathcal{C}$. The following theorem characterizes the closure property of proper interval graphs under powers.

THEOREM 2.1. *[33] The class of proper interval graphs is strongly closed under powers.*

In particular, if H is a proper interval graph, then H^2 is a proper interval graph. In light of this theorem, to determine if G is the square of a proper interval graph, we can, without loss of generality, assume that G is a proper interval graph.

Given a graph G, there are $\mathcal{O}(n + m)$ algorithms (e.g. [5]) which determine if G is a proper interval graph and construct an interval representation if it is. Therefore, in the following sections, to determine if G is the square of a proper interval graph, we assume G is a proper interval graph and the corresponding interval representation is given. If G is the square of a proper interval graph, we use H to denote a proper interval graph square root of G. We will let $V(G) = \{g_1, g_2, \ldots, g_n\}$ and $V(H) = \{h_1, h_2, \ldots, h_n\}$. We define $G[i, j] = G[g_i, g_{i+1}, g_{i+2}, \ldots, g_j]$ and similarly for H. Without loss of generality, we assume G and H are connected in the rest of this section.

2.1.1. Computing the square of a proper interval graph. Before we discuss how to find a proper interval square root H of a given proper interval graph G, we first consider how to compute H^2 from a proper interval graph H. This will give us insight into how to do the reverse operation. In general graphs, we can compute the square of a graph by doing matrix multiplication. But in proper interval graphs, there is a more effective and yet very intuitive way to compute the square of H by looking at the interval representation of H. Figure 2.1 presents a proper interval graph H and Figure 2.2 shows H^2. In the figures, numbers on the left represent the vertex names while numbers on the right represent the leftmost neighbor names (to be defined later). This example will be used throughout this section.

Given a total vertex ordering of a proper interval graph, the following properties are obvious.

PROPOSITION 2.2. *Given a proper interval graph H with a total vertex ordering, $N_H[h_i]$ is consecutive for any $1 \leq i \leq n$.*

With this property, it is easy to compute the square of a proper interval graph. By Proposition 2.2, we know that $N_H[h_j]$ is consecutive. Let $N_H[h_j] = \{h_i, \ldots, h_k\}$. We say h_i is the *leftmost neighbor* of h_j denoted by $l_H(h_j)$ and h_k is the *rightmost*

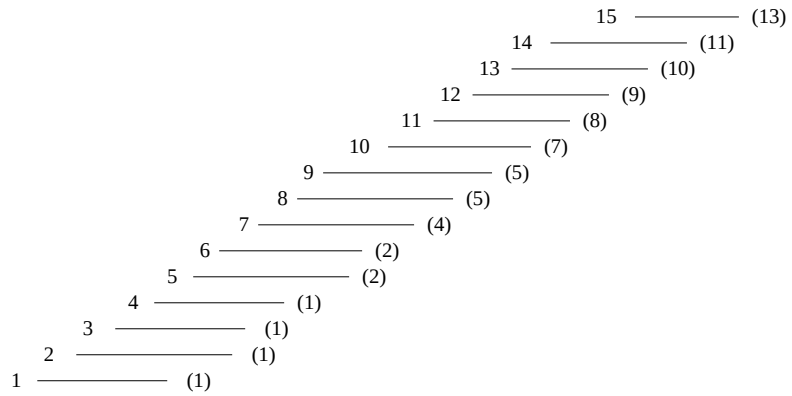

FIG. 2.1. *A proper interval graph H together with the names of the parents.*

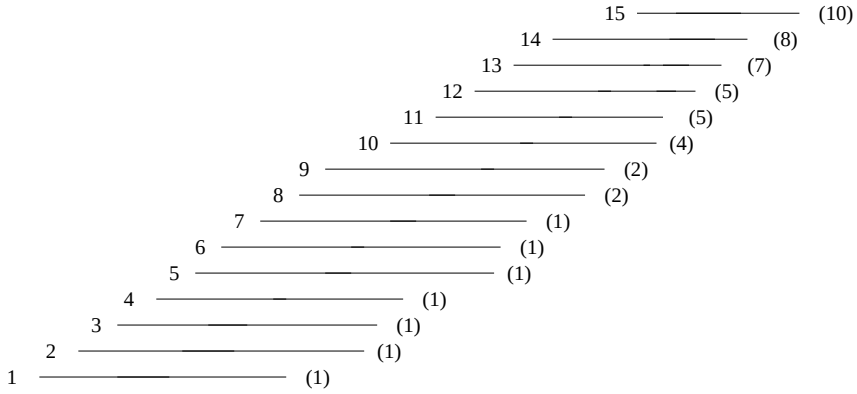

FIG. 2.2. $G = H^2$.

neighbor of h_j denoted by $r_H(h_j)$. For technical reasons, we set $l_H(h_1) = h_1$ and $r_H(h_n) = h_n$.

PROPOSITION 2.3. *Let $N_H[h_j] = \{h_i, \ldots, h_k\}$ with $1 \leq i \leq j \leq k \leq n$. Then $N_{H^2}[h_j] = N_H[h_i] \cup N_H[h_k]$ for any $1 \leq j \leq n$. In other words, $N_{H^2}[h_j] = N_H[l_H(h_j)] \cup N_H[r_H(h_j)]$.*

COROLLARY 2.4. *Given a proper interval graph H, $N_H^2[h_j] = \{l_H(l_H(h_j)), \ldots, r_H(r_H(h_j))\}$ for any $1 \leq j \leq n$.*

Notice that if $i < j$, then $left(l(l(h_i))) \leq left(l(l(h_j)))$ and $right(r(r(h_i))) \leq right(r(r(h_j)))$. So a total vertex ordering in H is a total vertex ordering in H^2. In other words, if G is the square of a proper interval graph, then it has a proper interval graph square root H such that G and H have the same vertex ordering. We will prove this fact formally later and it is very useful when we construct a proper interval graph square root since it significantly reduces the search space.

2.1.2. Notation for the algorithm. We introduce some notation to facilitate our discussion. We say v_i is the *parent* of v_j and v_j is a *child* of v_i if v_i is the leftmost neighbor of v_j. In Figure 2.1 and Figure 2.2, the number in parenthesis beside an interval indicates the parent of the corresponding vertex. Notice that if v_j is a child of v_i, then v_j is not adjacent to v_{i-1}. For every v_i, there is a unique parent, denoted by $p(v_i)$. On the other hand, v_i may have many children and we denote the set of

children of v_i by $C(v_i)$. For example, in Figure 2.1, $C_H(h_2) = \{h_5, h_6\}$. For technical reasons, $C(v_1) = N(v_1)$ (i.e. $v_1 \notin C(v_1)$) but $p(v_1) = v_1$. Note that v_i may have no children. For example, in Figure 2.1, $C_H(h_3) = \emptyset$. If $v_i = l(l(v_j))$, then we say v_i is the *grandparent* of v_j, denoted by $gp(v_j)$, and v_j is a *grandchild* of v_i.

Let X be a set of consecutive vertices. We define $C(X) = \bigcup_{v \in X} C(v)$. Notice that it is possible that $X \cap C(X) \neq \emptyset$; however, we are only interested in the case when $X \cap C(X) = \emptyset$ and thus $X < C(X)$. $C(X)$ is the union of the set of children of vertices in X. We let $C^0(X) = X$, $C^1(X) = C(X)$, and more generally $C^i(X) = C(C^{i-1}(X))$. Notice that if $C^i(X) = \emptyset$, then $C^{i+1}(X) = \emptyset$. We define $e(X) = k$ where k is the maximum value such that $C^k(X) \neq \emptyset$ and we define $C^*(X) = \bigcup_{0 \leq i \leq e(X)} C^i(X)$. So $C^*(X)$ is the set of descendants of vertices in X together with X. We say that $C^*(X)$ is the *chain* of X and $e(X)$ is the *length* of the chain. If $v' \in C^1(v)$, then v' is a child of v, and if $v' \in C^2(v)$, then v' is a grandchild of v. Furthermore if $v' \in C^*_G(v)$, then v' is a descendant of v. Finally, if $X < Y < C_G(X)$, then we denote $C^*_G(X) \cup C^*_G(Y)$ by $C^*_G(X, Y)$.

The remainder of this section goes as follows. In subsection 2.2, we will prove that if G is a proper interval graph square, then there exists a proper interval graph H with the same vertex ordering as in G such that $H^2 = G$. Then we will show some properties of chains in subsection 2.3, and prove the decomposition theorem in subsection 2.4. Finally, we present the algorithm and analyze its complexity in subsection 2.5 and extend it to recognize k-th powers of proper interval graphs in subsection 2.6.

2.2. H^2 and H share the same vertex ordering.

LEMMA 2.5. *If G is a proper interval graph square, then there exists a proper interval graph H with the same vertex ordering as in G such that $H^2 = G$.*

Proof. We prove by induction on i that there exists H_i such that $\{g_1, \ldots, g_i\}$ in G are mapped to $\{h_1, \ldots, h_i\}$ in H_i and ${H_i}^2 = G$. And thus $H' = H_n$ is a proper interval graph with the same vertex ordering as in G.

First we prove the base case when $i = 1$. Suppose H is a proper interval graph square root of G. If g_1 in G is mapped to h_1 in H, then $H_1 = H$ and we are done. So suppose g_1 in G is mapped to h_j in H such that $1 < j$; we consider two cases.

Case 1: In H, there are $a < j < b$ such that $d_H(h_a, h_j) = 2$ and $d_H(h_j, h_b) = 2$. In H^2, h_j is adjacent to h_a and h_b. Since $I_G(g_1)$ is the leftmost interval in G, g_1 is a simplicial vertex in G and its neighborhood induces a clique in G. Since $G = H^2$, h_j is a simplicial vertex in H^2 and thus h_a and h_b are adjacent in H^2. Since $h_j h_a, h_j h_b \notin E(H)$, $right_H(h_a) < left_H(h_j)$ and $right_H(h_j) < left_H(h_b)$ in H. Therefore, h_a and h_b are not adjacent in H. Since h_a and h_b are adjacent in H^2, there exists h_c such that $h_a h_c \in E(H)$ and $h_b h_c \in E(H)$. However, this implies that $left_H(h_c) < right_H(h_a) < left_H(h_j) < right_H(h_j) < left_H(h_b) < right_H(h_c)$ and thus $I_H(h_c)$ is an interval which properly contains $I_H(h_j)$. This contradicts the assumption that H is a proper interval graph. So case 1 is impossible.

Case 2: In H, $I_H(h_j)$ either intersects all the intervals on its left or it intersects all the intervals on its right. Without loss of generality, we assume the former case such that h_i is a neighbor of h_j for all $i < j$. As $I_G(g_1)$ is the leftmost interval in G, g_1 is a simplicial vertex and $N_G[g_1] \subseteq N_G[g_k]$ if g_k is a neighbor of g_1 in G. Since $G = H^2$ and g_1 is mapped to h_j, $N_{H^2}[h_j] \subseteq N_{H^2}[h_i]$ for all $i < j$. On the other hand, by Corollary 2.4, $right_H(h_j) > right_H(h_i)$ implies $N_{H^2}[h_i] \subseteq N_{H^2}[h_j]$ for all $i < j$. Therefore $N_{H^2}[h_i] = N_{H^2}[h_j]$ for all $i < j$. In particular, $N_{H^2}[h_1] = N_{H^2}[h_j]$. Therefore, we can construct H_1 from H by switching the preimages of h_1 and h_j.

Clearly, ${H_1}^2 = H^2 = G$. So the base case holds.

Now we assume $g_1, \ldots, g_k$ in G are mapped to $h_1, \ldots, h_k$ in H_k respectively and ${H_k}^2 = G$. We will construct H_{k+1} from H_k such that $g_1, \ldots, g_{k+1}$ in G are mapped to $h_1, \ldots, h_{k+1}$ in H_{k+1} respectively and ${H_{k+1}}^2 = G$. If g_{k+1} in G is mapped to h_{k+1} in H_k, then $H_{k+1} = H_k$ and we are done. So suppose g_{k+1} in G is mapped to h_j in H_k with $j > k+1$. Since $I_G(g_{k+1})$ is the leftmost interval in $G[k+1, n]$, by Corollary 2.4, we have $N_{G[1,k]}[g_i] \subseteq N_{G[1,k]}[g_{k+1}]$ for $i > k+1$. Since $G = H_k^2$ and g_{k+1} is mapped to h_j, $N_{H_k{}^2[1,k]}[h_i] \subseteq N_{H_k{}^2[1,k]}[h_j]$ for $k+1 \leq i < j$. On the other hand, since $left_{H_k}(h_i) < left_{H_k}(h_j)$ for any $k+1 \leq i < j$, by Corollary 2.4, $N_{H_k{}^2[1,k]}[h_j] \subseteq N_{H_k{}^2[1,k]}[h_i]$ for $k+1 \leq i < j$. So, $N_{H_k{}^2[1,k]}[h_i] = N_{H_k{}^2[1,k]}[h_j]$ for $k+1 \leq i < j$. By essentially the same argument, we also have $N_{H_k{}^2[k+1,n]}[h_j] = N_{H_k{}^2[k+1,n]}[h_i]$ for $k+1 \leq i < j$.

Therefore $N_{H_k{}^2}[h_i] = N_{H_k{}^2}[h_j]$ for $k+1 \leq i < j$. In particular, $N_{H_k{}^2}[h_{k+1}] = N_{H_k{}^2}[h_j]$. Thus we may construct H_{k+1} from H_k by switching the preimages of h_j and h_{k+1}. Clearly, ${H_{k+1}}^2 = H_k^2 = G$ and $g_1, \ldots, g_{k+1}$ in G are mapped to $h_1, \ldots, h_{k+1}$ in H_{k+1}. By induction, we can construct H_n. So $H' = H_n$ and this completes the proof. □

By Lemma 2.5, we can assume that if G is a proper interval graph square, then there is a proper interval graph H such that $H^2 = G$ and H and G share the same vertex ordering (i.e. g_i in G is mapped to h_i in H for all i). So later on, g_i and h_i actually mean the same vertex but g_i refers to the one in G and h_i refers to the one in H.

2.3. Parent and children relationship. Now we characterize H based on the parent-children relationship in G which is analogous to Corollary 2.4. But in the following lemma, we characterize H by just using the leftmost neighbors instead of using both leftmost and rightmost neighbors. This allows us to construct H by considering one direction only (from left to right, i.e. from 1 to n).

LEMMA 2.6. *Let G be a proper interval graph square and let H be a proper interval graph with the same vertex ordering as in G. $H^2 = G$ if and only if $gp_H(h_j) = p_G(g_j)$ for $1 \leq j \leq n$.*

Proof. Since G and H share the same vertex ordering, if $H^2 = G$, by Corollary 2.4, $gp_H(h_j) = p_G(g_j)$ for $1 \leq j \leq n$.

Now we prove the reverse direction. If $gp_H(h_j) = p_G(g_j)$ for $1 \leq j \leq n$, we prove that $H^2 = G$ by induction. First we prove the base case that $N_{H^2}[h_n] = N_G[g_n]$. By Corollary 2.4, $N_{H^2}[h_n] = \{l_H(l_H(h_n)), \ldots, h_n\}$. Since $l_H(l_H(h_n)) = gp_H(h_n) = p_G(g_n)$ and G and H share the same vertex ordering, $N_{H^2}[h_n] = N_G[g_n]$ and thus the base case holds.

Now suppose that $N_{H^2}[h_j] = N_G[g_j]$ for $k+1 \leq j \leq n$. Consider $N_{H^2}[h_k]$; by the induction hypothesis, $N_{H^2}[h_k] \cap \{h_{k+1}, \ldots, h_n\} = N_G[g_k] \cap \{g_{k+1}, \ldots, g_n\}$. Since $l_H(l_H(h_k)) = gp_H(h_k) = p_G(g_k)$ and G and H share the same vertex ordering, $N_{H^2}[h_k] \cap \{h_1, \ldots, h_k\} = N_G[g_k] \cap \{g_1, \ldots, g_k\}$. Therefore, we conclude that $N_{H^2}[h_k] = N_G[g_k]$ and this completes the induction step. □

The following easily proved proposition describes a basic property of a chain based on the definition.

PROPOSITION 2.7. *If $X < Y < C(X)$, then $C^i(X) < C^i(Y) < C^{i+1}(X)$ for $i \geq 0$.*

The following lemma formalizes the idea that a chain in H is composed of two chains in G.

LEMMA 2.8. *Let $H^2 = G$ where H and G share the same vertex ordering. If*

$X < Y < C_G(X)$ and $C_H(X) = Y$, then $C_H(C_G^i(X)) = C_G^i(Y)$ and $C_H(C_G^i(Y)) = C_G^{i+1}(X)$ for $i \geq 0$. Furthermore, $e_G(X) = e_G(Y)$ or $e_G(X) = e_G(Y) + 1$.

Proof. Since $X < Y < C_G(X)$ and $C_H(X) = Y$, by Lemma 2.6, $C_H(Y) = C_G(X)$. And since $H^2 = G$, if $C_G(X) \neq \emptyset$, then $Y \neq \emptyset$. By Proposition 2.7, $C_G(X) < C_G(Y)$. Repeating the same argument, since $Y < C_G(X) < C_G(Y)$ and $C_H(Y) = C_G(X)$, we have $C_H(C_G(X)) = C_G(Y)$. And since $H^2 = G$, if $C_G(Y) \neq \emptyset$, then $C_G(X) \neq \emptyset$. By induction, we have $C_H(C_G^i(Y)) = C_G^{i+1}(X)$ and $C_H(C_G^i(X)) = C_G^i(Y)$. Also, by Lemma 2.6, if $C_G^{i+1}(X) \neq \emptyset$, then $C_G^i(Y) \neq \emptyset$; if $C_G^{i+1}(Y) \neq \emptyset$, then $C_G^{i+1}(X) \neq \emptyset$. So $e_G(X) = e_G(Y)$ or $e_G(X) = e_G(Y) + 1$ and this completes the proof. □

So, a necessary condition for $C_G^*(X,Y)$ to form $C_H^*(X)$ is that the length of $C_G^*(X)$ and the length of $C_G^*(Y)$ are about the same.

The following two propositions are useful for decomposition. Their proofs follow directly from the definitions.

PROPOSITION 2.9. *Suppose $X < C(X)$. If $X_1 \cup X_2 = X$ and $X_1 < X_2$, then $C^i(X_1) \cup C^i(X_2) = C^i(X)$ and $C_G^i(X_1) < C_G^i(X_2)$ for $i \geq 0$.*

PROPOSITION 2.10. *Suppose $X < Y < C(X)$. Let $X_1 \cup X_2 = X$ and $X_1 < X_2$ and similarly $Y_1 \cup Y_2 = Y$ and $Y_1 < Y_2$. Then $C^i(X) = C^i(X_1) \cup C^i(X_2)$, $C^i(X_1) < C^i(X_2)$ and $C^i(Y) = C^i(Y_1) \cup C^i(Y_2)$, $C^i(Y_1) < C^i(Y_2)$ for $i \geq 0$. Also, $C^i(X_1) < C^i(X_2) < C^i(Y_1) < C^i(Y_2) < C^{i+1}(X_1)$ for $i \geq 0$.*

Proof. The first two statements follow from Proposition 2.9. The last statement follows from Proposition 2.7. □

2.4. A decomposition theorem. The following is an important definition for our algorithm. Intuitively, it checks if $C_G^*(X,Y)$ can form $C_H^*(X)$; in other words, to check if $G[C_G^*(X,Y)]$ has a square root with special properties.

DEFINITION 2.11. *Suppose $X < Y < C_G(X)$; $G[C_G^*(X,Y)]$ is* matched *if and only if there exists H' with the same vertex ordering as $G[C_G^*(X,Y)]$ such that*

(P1) $H'^2 = G[C_G^*(X,Y)]$;

(P2) $H'[X]$ is a clique;

(P3) $p_{H'}(h_i) \in X$ if and only if $h_i \in X \cup Y$.

We say H' is a matched root *of $G[C_G^*(X,Y)]$.*

We will decompose G into small graphs that correspond to chains in H, then we will construct matched roots (i.e. chains in H) independently and combine them to form a root of G. Note that (P2) and (P3) are necessary conditions for H' to be $H[C_H^*(X)]$.

The following lemma is a rephrasing of Lemma 2.6 in the case of matched roots. It characterizes matched roots based on the parent-children relationship in G.

LEMMA 2.12. *Suppose $X < Y < C_G(X)$; let H have the same vertex set and the same vertex ordering as $G[C_G^*(X,Y)]$ and assume H satisfies (P2) and (P3). Then $H^2 = G[C_G^*(X,Y)]$ if and only if $gp_H(h_i) = p_G(g_i)$ for all $g_i \in C_G^*(X,Y) - X - Y$.*

Proof. By Lemma 2.6, $H^2 = G[C_G^*(X,Y)]$ if and only if $gp_H(h_i) = p_{C_G^*(X,Y)}(g_i)$ for all $g_i \in C_G^*(X,Y)$. Notice that $p_{C_G^*(X,Y)}(g_i) = p_G(g_i)$ for all $g_i \in C_G^*(X,Y) - X$. Also, (P2) and (P3) holding for H guarantee $gp_H(h_i) = p_G(g_i)$ for all $g_i \in X \cup Y$. Therefore, $H^2 = G[C_G^*(X,Y)]$ if and only if $gp_H(h_i) = p_G(g_i)$ for all $g_i \in C_G^*(X,Y) - X - Y$. □

The following is a rephrasing of Lemma 2.8 in the case of a matched root. Note that the only difference between Lemma 2.13 and Lemma 2.8 is that $C_H(C_G^0(X)) = C_G^0(Y)$ holds in Lemma 2.8.

LEMMA 2.13. *Suppose $G[C_G^*(X,Y)]$ is matched and let H be a matched root of $G[C_G^*(X,Y)]$. Then $C_H(C_G^i(X)) = C_G^i(Y)$ for $i \geq 1$ and $C_H(C_G^i(Y)) = C_G^{i+1}(X)$ for*

$i \geq 0$. *Furthermore,* $e_G(X) = e_G(Y)$ *or* $e_G(X) = e_G(Y) + 1$.

Proof. By (P3) of H, $p_H(h_i) \in X$ if and only if $h_i \in X \cup Y$. Since $H^2 = G[C^*_G(X,Y)]$, $C_H(Y) = C_G(X)$. By Proposition 2.7, $C_G(X) < C_G(Y)$. Since $C_H(Y) < C_G(X) < C_G(Y)$ and $C_H(Y) = C_G(X)$, by Lemma 2.8, the results follow. □

As mentioned previously, if $H^2 = G$, then we can combine matched roots to construct root of G. The following lemma shows the reverse direction, namely, we can find matched roots in H. This justifies our approach of constructing matched roots.

LEMMA 2.14. *Let* H *have the same vertex ordering as in* G *and* $H^2 = G$. *If* $X < Y < C_G(X)$, $H[X]$ *is a clique and* $C_H(X) = Y$, *then* $G[C^*_G(X,Y)]$ *is matched.*

Proof. Let $H' = H[C^*_H(X)]$; we will show that H' is a matched root of $G[C^*_G(X,Y)]$. First, since $H^2 = G$, $X < Y < C_G(X)$ and $C_H(X) = Y$, by Lemma 2.8, $C^*_H(X) = C^*_G(X,Y)$. We now show (P1)-(P3) are satisfied for H'. Since $H[X]$ is a clique, (P2) is satisfied. Since $C_H(X) = Y$, (P3) is satisfied. For (P1), by our construction, $p_{H'}(h_i) = p_H(h_i)$ for all $h_i \in C^*_H(X) - X$. So in H', $gp_{H'}(h_i) = gp_H(h_i)$ for all $h_i \in C^*_H(X) - X - C_H(X)$. Since $H^2 = G$, by Lemma 2.6, $gp_H(h_i) = p_G(g_i)$ for all i. Therefore $gp_{H'}(h_i) = p_G(g_i)$ for all $g_i \in C^*_G(X,Y) - X - Y$. By Lemma 2.12, $H'^2 = G[C^*_G(X,Y)]$ and thus (P1) is satisfied; by Definition 2.11, $G[C^*_G(X,Y)]$ is matched. □

Now we have enough machinery to prove the decomposition theorem. First, the following theorem reduces the original problem to finding a matched root. Intuitively, it corresponds to the partition of the neighborhood of g_1 in G (i.e. guessing the neighborhood of h_1 in H).

THEOREM 2.15. G *is a proper interval graph square if and only if* $G[C^*_G(X,Y)]$ *is matched for some* X *where* $X \cup Y = C_G(g_1)$ *and* $X < Y$.

Proof. If G is a proper interval graph square, by Lemma 2.5, then there exists H with the same vertex ordering as in G and $H^2 = G$. Let $X = C_H(h_1)$ and $Y = C_G(g_1) - X$; clearly $X \cup Y = C_G(g_1)$ and $X < Y$. We will show that $G[C^*_G(X,Y)]$ is matched. In H, since $X = C_H(h_1)$, $H[X]$ is a clique. Since $Y = C_G(g_1) - X$ and $H^2 = G$, every vertex in Y has a neighbor in X and thus $C_H(X) = Y$. So $H[X]$ is a clique, $C_H(X) = Y$ and $X < Y < C_G(X)$; by Lemma 2.14, $G[C^*_G(X,Y)]$ is matched.

Now suppose $G[C^*_G(X,Y)]$ is matched for some X where $X \cup Y = C_G(g_1)$ and $X < Y$. Let H' be a matched root of $G[C^*_G(X,Y)]$. We show how to construct H from H' such that $H^2 = G$. Since $X \cup Y = C_G(g_1)$, by Proposition 2.9, $C^*_G(X,Y) = C^*_G(C_G(g_1))$. Since $C_G(g_1) = N_G(g_1)$, $C^*_G(C_G(g_1)) = G - g_1 = \{g_2, \ldots, g_n\}$. Since H' is a matched root of $G[C^*_G(X,Y)]$, $V(H') = \{h_2, \ldots, h_n\}$. Since $H'^2 = G - g_1$, $H'[X]$ is a clique and $C_{H'}(X) = Y$, by constructing $H = H' + h_1$ with $C_H(h_1) = X$, it is easy to verify that $H^2 = G$. So G is a proper interval graph square. □

The following theorem is the core of the algorithm. It reduces the problem of finding a matched root in a graph to finding matched roots in two smaller graphs. It is the basis of the "decomposition" step in the algorithm.

THEOREM 2.16. *Given* X *and* Y *with* $X < Y < C_G(X)$ *in* G *with* $|X| \geq 2$. *Let* g_a *be the first vertex in* X. *Then* $G[C^*_G(X,Y)]$ *is matched if and only if* $G[C^*_G(g_a, A)]$ *and* $G[C^*_G(X - g_a, Y - A)]$ *are both matched for some* A *where* $A < Y - A$. *(Note:* A *could be an emptyset.)*

Proof. Suppose $G[C^*_G(X,Y)]$ is matched and let H' be a matched root of $G[C^*_G(X,Y)]$. We will show $G[C^*_G(g_a, A)]$ and $G[C^*_G(X - g_a, Y - A)]$ are both matched for some A where $A < Y - A$. In particular, we set $A = C_{H'}(h_a) - X$. Since H' has the same ver-

tex ordering as $G[C^*_G(X,Y)]$, $A < Y - A$. If $C_G(X) = \emptyset$, by Lemma 2.13, $C^*_G(X,Y) = X \cup Y$ and clearly $G[C^*_G(g_a, A)]$ and $G[C^*_G(X - g_a, Y - A)]$ are both matched. So we assume $C_G(X) \neq \emptyset$; by Lemma 2.13, $C_{H'}(Y) = C_G(X)$. Since ${H'}^2 = G[C^*_G(X,Y)]$, by our choice of A, $C_{H'}(A) = C_G(g_a)$ and $C_{H'}(Y - A) = C_G(X - g_a)$. By Lemma 2.14, both $G[C^*_G(A, C_G(g_a))]$ and $G[C^*_G(Y - A, C_G(X - g_a))]$ are matched and we let H_1 and H_2 be the corresponding matched roots.

We will show that $H'_1 = H'[H_1 + h_a]$ and $H'_2 = H'[H_2 + (X - h_a)]$ are matched roots of $G[C^*_G(g_a, A)]$ and $G[C^*_G(X - g_a, Y - A)]$ respectively. By (P2) for H', $H'[X]$ is a clique, so $H'[h_a]$ and $H'[X - h_a]$ are cliques and thus (P2) is satisfied in H'_1 and H'_2. By (P3) for H', $p_{H'}(h_i) \in X$ if and only if $h_i \in X \cup Y$. Since we set $A = C_{H'}(h_a) - X$, it is clear that (P3) is satisfied in H'_1 and H'_2. By (P1) for H_1 and H_2, ${H_1}^2 = G[C^*_G(A, C_G(g_a))]$ and ${H_2}^2 = G[C^*_G(Y - A, C_G(X - g_a))]$. Since H' is a matched root of $G[C^*_G(X,Y)]$, it is easy to check that ${H'_1}^2 = G[C^*_G(g_a, A)]$ and ${H'_2}^2 = G[C^*_G(X - g_a, Y - A)]$. Thus, by Definition 2.11, $G[C^*_G(g_a, A)]$ and $G[C^*_G(X - g_a, Y - A)]$ are both matched.

Suppose $G[C^*_G(g_a, A)]$ and $G[C^*_G(X - g_a, Y - A)]$ are both matched for some A where $A < Y - A$ and let H_1 and H_2 be the corresponding matched roots. We will show how to construct a matched root H' of $G[C^*_G(X,Y)]$. By Proposition 2.10, $C^*_G(g_a, A) \cup C^*_G(X - g_a, Y - A) = C^*_G(X,Y)$. Also by Proposition 2.10, we know the total vertex order of $H_1 \cup H_2$. We can construct H' by combining H_1 and H_2 with the order indicated by Proposition 2.10 and setting $p_{H'}(h_i) = h_a$ for $h_i \in X - h_a$. Notice that in this ordering, if $i \leq j$, then $left(p_{H'}(h_i)) \leq left(p_{H'}(h_j))$. So H' is constructible with the parent-children relationship unchanged as in H_1 and H_2, except for $h_i \in X - h_a$. The concept is shown in Figure 2.3 where we combine two matched roots to a matched root, recall that a matched root corresponds to two chains in G. It is important to note that, in Figure 2.3, the parent-children relationship in H' unchanged as in H_1 and H_2 (by looking at the intersection of intervals), except for $h_i \in X - h_a$. It is also worth noting that, unlike Figure 2.3, the length of $C^*_G(g_a, A)$ and the length of $C^*_G(X - g_a, Y - A)$ could be very different.

By Lemma 2.12, $gp_{H_1}(h_i) = p_G(g_i)$ for all $g_i \in C^*_G(g_a, A) - g_a - A$ and $gp_{H_2}(h_j) = p_G(g_j)$ for all $g_j \in C^*_G(X - g_a, Y - A) - (X - g_a) - (Y - a)$. As the parent relationship in H' is unchanged as in H_1 and H_2, except for vertices in $X - g_a$, so $gp_{H'}(h_i) = p_G(g_i)$ for all $g_i \in C^*_G(X,Y) - X - Y$. Since we set $p_{H'}(h_i) = h_a$ for $h_i \in X - g_a$, $H'[X]$ is a clique and thus H' satisfies (P2). By (P3) for H_1 and H_2, $p_{H_1}(h_i) \in \{h_a\}$ if and only if $h_i \in \{h_a\} \cup A$ and $p_{H_2}(h_j) \in X - h_a$ if and only if $h_j \in (X - h_a) \cup (Y - A)$. Without changing the parent relationship except for vertices in $X - g_a$, H' satisfies (P3). By Lemma 2.12, ${H'}^2 = G[C^*_G(X,Y)]$ and thus $G[C^*_G(X,Y)]$ is matched. □

The following theorem is the basis of the "propagation" step. It reduces the problem of finding a matched root in a graph to finding a matched root in a smaller graph; in particular, it reduces the length of the chain by one.

THEOREM 2.17. *Given $X = \{g_i\}$ and Y with $\{g_i\} < Y < C_G(g_i)$. Then $G[C^*_G(\{g_i\}, Y)]$ is matched if and only if $G[C^*_G(Y, C_G(g_i))]$ is matched.*

Proof. Suppose $G[C^*_G(\{g_i\}, Y)]$ is matched and let H' be a matched root of $G[C^*_G(\{g_i\}, Y)]$. We will show that $H' - h_i$ is a matched root of $G[C^*_G(Y, C_G(g_i))]$. By (P3) for H', h_i is the parent of the vertices in Y in H'. So $H'[Y]$ is a clique and thus (P2) is satisfied in $H' - h_i$. Also, by Lemma 2.13, $C_{H'}(Y) = C_G(g_i)$ and thus (P3) is satisfied in $H' - h_i$. And clearly, $(H' - h_i)^2 = G - g_i$. By Definition 2.11, $G[C^*_G(Y, C_G(g_i))]$ is matched.

Suppose $G[C^*_G(Y, C_G(g_i))]$ is matched and let H_1 be a corresponding matched

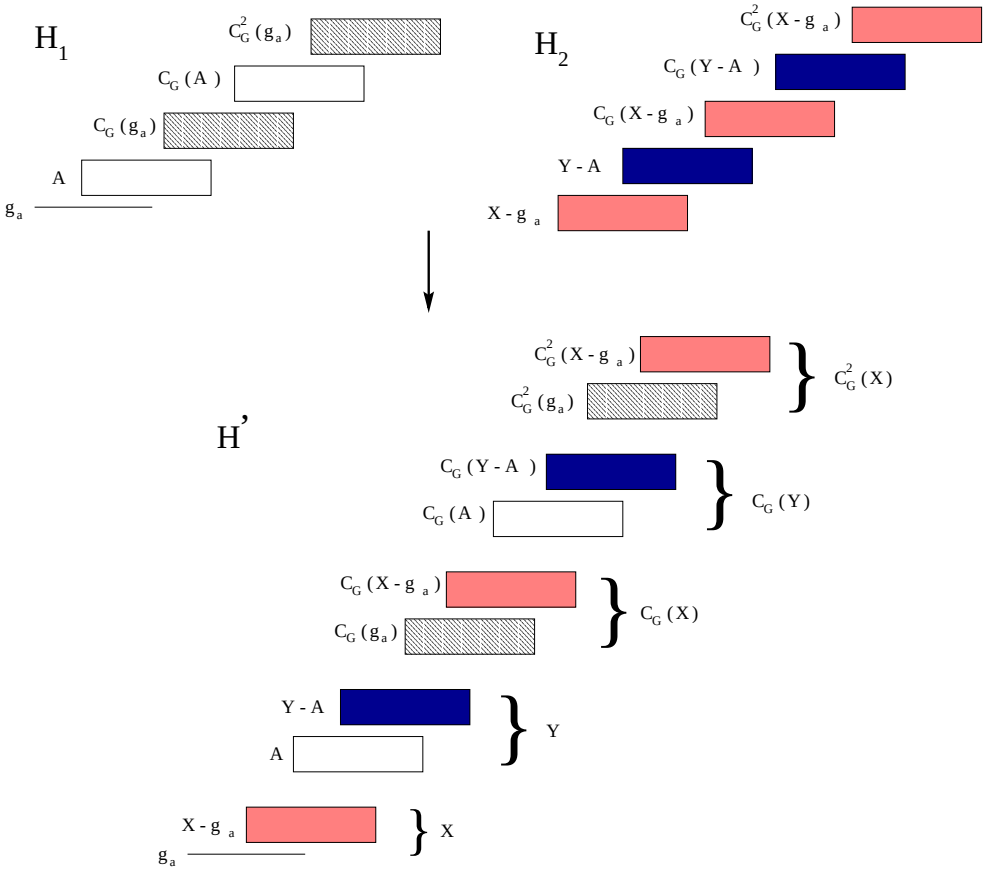

FIG. 2.3. *Combine matched root H_1 of $C_G^*(g_a, A)$ and matched root H_2 of $C_G^*(X - g_a, Y - A)$ into matched root H' of $C_G^*(X, Y)$.*

root. By (P2), $H_1[Y]$ is a clique. By (P3), $p_{H_1}(h_j) \in Y$ if and only if $h_j \in Y \cup C_G(g_i)$. By constructing $H' = H_1 + h_i$ with $C_H(h_i) = Y$, ${H'}^2 = G[C_G^*(\{g_i\}, Y)]$ and (P2), (P3) are satisfied in H'. So, by Definition 2.11, $G[C_G^*(\{g_i\}, Y)]$ is matched. □

2.5. Algorithm, correctness and complexity. The following recursive algorithm is an implementation of the decomposition procedure. P is a *prefix* of S if $P < S - P$, note that P could be an emptyset.

```
MAIN PROGRAM
    for every prefix P of C_G(g_1) do
        if MATCH (P, C_G(g_1) - P) then output "YES"
    output "NO"
```

```
MATCH (X = {g_a, ...}, Y)
    Case: X = ∅
        if Y = ∅ then return TRUE
        else return FALSE
    Case: X = {g_a}
        return MATCH (Y, C_G(g_a))
    Otherwise:
        for every prefix P of Y do
            if ( MATCH ({g_a}, P) and MATCH (X - g_a, Y - P) )
            then return TRUE
        return FALSE
```

THEOREM 2.18. *The algorithm is correct.*

Proof. In MAIN PROGRAM, we reduce the recognition of the square of a proper interval graph to finding a matched root. Then we use the results obtained in the previous subsection to find a matched root; this is done by the recursive function MATCH.

The correctness of the MAIN PROGRAM is justified by Theorem 2.15. Now we look into the recursive function MATCH. The first case is the base case when the X-chain ends. If the Y-chain also ends, then X and Y are matched. Otherwise, X and Y are not matched. The second case is the "propagation" step. Its correctness is justified by Theorem 2.17. The final case is the "decomposition" step. Its correctness is justified by Theorem 2.16. □

THEOREM 2.19. SQUARE OF PROPER INTERVAL GRAPH *can be solved in* $\mathcal{O}(n^5)$.

Proof. The key observation is that $X = \{g_a, g_{a+1}, \ldots, g_b\}$ and $Y = \{g_c, g_{c+1}, \ldots, g_d\}$ are consecutive sets. So there are at most $\mathcal{O}(n^4)$ instances of MATCH since there are at most n possibilities of each a, b, c, d. Therefore, we can use dynamic programming to solve this problem. Explicitly, we can create a table of size $\mathcal{O}(n^4)$ to store the results of all possible instances. Each entry in the table can be computed in time at most $\mathcal{O}(n)$. The total complexity is at most $\mathcal{O}(n^5)$. To implement the algorithm, we can use a standard trick of "caching" the solutions when we run the recursion so that each entry of the table is computed at most once. □

Notice that the above algorithm is for the decision problem where a proper interval graph is a proper interval graph square. To actually find a proper interval graph square root, we need to add another four dimensional array to trace the partitions. This is a standard technique of dynamic programming and is covered in many textbooks. For simplicity in the description of the algorithm, we do not include the details of finding the partitions.

Now suppose we know the partitions; we show how to construct a proper interval square root H of G. Suppose $X = \{h_a, \ldots, h_b\}$ and $Y = \{h_c, \ldots, h_d\}$ and the partition is $(h_a, \{h_c, \ldots, h_i\})$ and $(X - h_a, \{h_{i+1}, \ldots, h_d\})$. Then we set $C_H(h_a) = \{h_c, \ldots, h_i\}$. Suppose $X = \{h_a\}$ and $Y = \{h_c, \ldots, h_d\}$; then we set $C_H(h_a) = Y$. So from the partitions, we can deduce the parent for any vertex. Then from the parents, we can obtain the adjacency matrix of a proper interval graph square root.

2.6. Outline of the recognition algorithm of k-th powers of proper interval graphs. By applying the same idea of the recognition algorithm of the square of a proper interval graph, we can develop a polynomial time algorithm for the recognition of the k-th power of a proper interval graph for any fixed k. We will not go into full details. Also, we assume that G is not a complete graph. The outline of the algorithm is as follows:

```
MAIN PROGRAM
    for every partition of C_G(g_1) into k consecutive sets S_1 < ... < S_k
        if MATCH (S_1, S_2, ..., S_k) then output "YES"
    output "NO"
```

```
MATCH (S_1 = {g_a, ...}, S_2, ..., S_k)
    Case: S_1 = ∅
        if S_2, ..., S_k are all emptyset then return TRUE
        else return FALSE
    Case: S = {g_a}
        return MATCH (S_2, ..., S_k, C_G(g_a))
    Otherwise:
        for every combination of prefixes P_2, ..., P_k of S_2, ..., S_k do
            if (MATCH ({g_a}, P_2, ..., P_k) and
                MATCH (S_1 - g_a, S_2 - P_2, ..., S_k - P_k))
            then return TRUE
        return FALSE
```

THEOREM 2.20. *k-TH POWER OF PROPER INTERVAL GRAPH can be solved in time $\mathcal{O}(n^{3k-1})$ and space $\mathcal{O}(n^{2k})$.*

Proof. To apply dynamic programming, we have to create a table of size $\mathcal{O}(n^{2k})$ since there are k pairs of integers. The main program is of complexity $\mathcal{O}(n^{k-1})$. In MATCH, it takes at most $\mathcal{O}(n^{k-1})$ to compute one entry in the table. The total complexity is at most $\mathcal{O}(n^{3k-1})$. □

3. NP-completeness. In this section, we will show that recognizing squares of chordal graphs, finding square roots of chordal graphs, and recognizing squares of split graphs are all NP-complete. We will use SET SPLITTING and INTERSECTION GRAPH BASIS as formulated in [12] for our reductions.

PROBLEM	[SP4] SET SPLITTING
INSTANCE	Collection C of finite sets of elements from S.
QUESTION	Is there 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?
NOTE	It is also known as HYPERGRAPH 2-COLORABILITY.

PROBLEM	[GT59] INTERSECTION GRAPH BASIS [19]
INSTANCE	Graph $G = (V, E)$, positive integer $K \leq \|E\|$.
QUESTION	Is G the intersection graph for a family of sets whose union has cardinality K or less, i.e., is there a K-element set S and for each $v \in V$ a subset $S[v] \subseteq S$ such that $\{u, v\} \in E$ if and only if $S[u]$ and $S[v]$ are not disjoint?

We will borrow the tail structure in [29] of a vertex v to ensure v has the same neighborhood in any square root H of G. It enables one to exactly pin down the neighborhood of v in any square root H of G.

LEMMA 3.1. *[29] If a, b, c, d are vertices of G such that*

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

Then the neighbors, in $V - \{a, b, c, d\}$, of d in any square root of G are the same as the neighbors, in $V - \{a, b, c, d\}$, of c in G (see Figure 3.1 for an illustration).

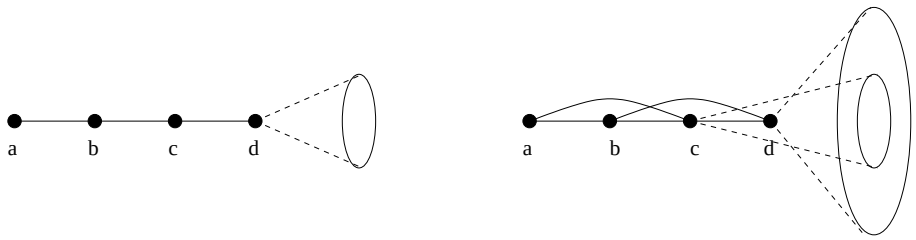

FIG. 3.1. *Tail in H and $G = H^2$.*

3.1. Squares of chordal graphs. In this subsection, we will show that to determine if G is the square of a chordal graph is NP-complete.

PROBLEM	SQUARE OF CHORDAL GRAPH
INSTANCE	A graph $G = (V, E)$.
QUESTION	Does there exist a *chordal graph H* such that $G = H^2$?

The rest of this section shows that SQUARE OF CHORDAL GRAPH is NP-hard by reducing SET SPLITTING to it. It is clear that SQUARE OF CHORDAL GRAPH is in NP, since guessing the square root H, verifying that H is a chordal graph and $G = H^2$ can easily be done in polynomial time. Thus we will conclude that SQUARE OF CHORDAL GRAPH is NP-complete.

3.1.1. The reduction. Given an instance of SET SPLITTING, we construct an instance of SQUARE OF CHORDAL GRAPH. Let c_j be the set of elements in subset j and let $C = \{c_1, \ldots, c_m\}$. 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 3.1):

Vertices of G

- *Element vertices:* U_i: $1 \leq i \leq n$ for each element u_i in S.
- *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.

Edges of G

- *Edges of tail vertices of subset vertices:* $\forall c_j \in C$ $C_j^3 \leftrightarrow C_j^2, C_j^3 \leftrightarrow C_j^1$,

$C_j^2 \leftrightarrow C_j^1, C_j^2 \leftrightarrow C_j$,
$C_j^1 \leftrightarrow C_j$ and $C_j^1 \leftrightarrow U_i$ for all $u_i \in c_j$.

- *Edges of subset vertices:* $\forall c_j \in C$
  $C_j \leftrightarrow S_1, C_j \leftrightarrow S_2, C_j \leftrightarrow U_i$ for all i and $C_j \leftrightarrow C_k$ iff $c_j \cap c_k \neq \emptyset$.
- *Edges of element vertices:* $\forall u_i \in U$
  $U_i \leftrightarrow U_j$ for all $j \neq i$ and $U_i \leftrightarrow S_1$ and $U_i \leftrightarrow S_2$.

Before presenting the details of the proof, we first give some intuition behind the transformation. From the tail structure, C_j^1 pins down the neighborhood of C_j in any square root H of G. So in any square root H of G, C_j is adjacent to U_i iff $u_i \in c_j$. Also, S_1 and S_2 are adjacent to all U_i in G to force S_1 and S_2 to have a common neighbor to all C_j in H. Moreover, S_1 and S_2 are not adjacent in G to force S_1 and S_2 to have no common neighbor in H and thus S_1, S_2 represents a partition of the ground set S.

LEMMA 3.2. *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 chordal graph H such that $H^2 = G$.*

Proof. Edges of H.

- *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 subset vertices:*
  $U_i \leftrightarrow U_j$ for $i \neq j$.

It is a tedious but straightforward task to check that $H^2 = G$. We leave the details to the reader (see [23] for a full proof). □

For example, given $C = \{c_1, c_2, c_3, c_4\}$, $c_1 = \{u_1, u_2, u_3\}, c_2 = \{u_2, u_5\}$, $c_3 = \{u_3, u_4\}$, $c_4 = \{u_1, u_4\}$ and $S = \{u_1, u_2, u_3, u_4, u_5\}$, we construct G as shown in Figure 3.2. The 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 indicates that each of them is universal 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 graph H corresponding to this solution is shown in Figure 3.3. The reader may verify that $H^2 = G$ and H is chordal. A possible perfect elimination ordering of H is $\{C_1^3, \ldots, C_4^3, C_1^2, \ldots, C_4^2, C_1^1, \ldots, C_4^1, C_1, \ldots, C_4, S_1, S_2, U_1, \ldots, U_5\}$.

We now show that if G is a square, 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. First, observing that $\{C_j^3, C_j^2, C_j^1, C_j\}$ satisfies the properties of Lemma 3.1, we have the following consequence:

PROPOSITION 3.3. *If H is a square root of G, then in H, C_j is adjacent only to: U_i if $u_i \in c_j$, and C_j^1.*

LEMMA 3.4. *If H is a square root of G, 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.*

Proof. Proposition 3.3 forces each subset vertex to be adjacent to its own elements only. Together with the fact that S_1 and S_2 are not adjacent to any tail vertices in G, S_1 and S_2 only have neighbors in the element set in H. Since $S_1 \nleftrightarrow S_2$ in G, they have no common element neighbor in H and so it is a partition on the element set. And since S_1 and S_2 are adjacent to all subset vertices in G but not in H, S_1 and S_2 have a common neighbor with C_i in the element set for all i. Therefore, $N_H(S_1)$ and $N_H(S_2)$ are the desired partitions. This completes the proof. □

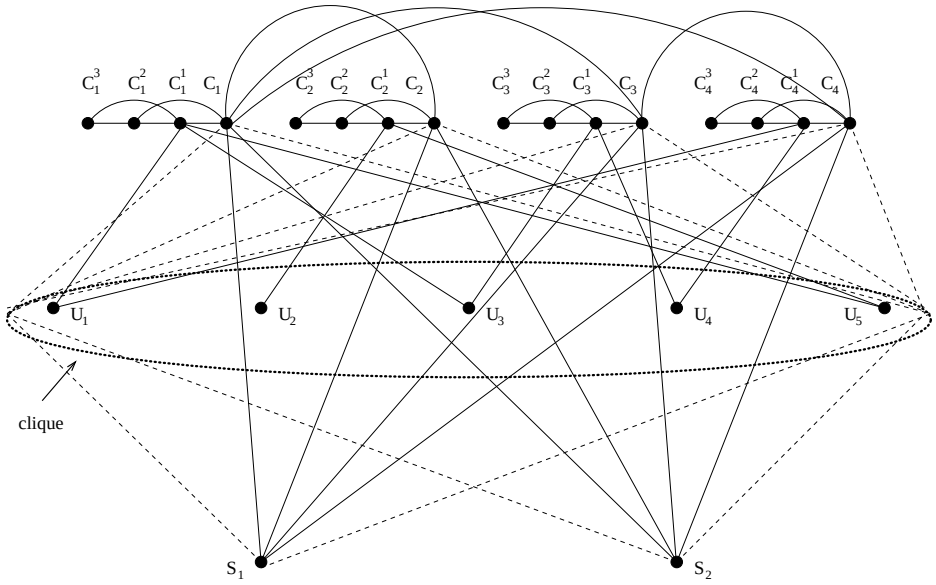

FIG. 3.2. *An example of G.*

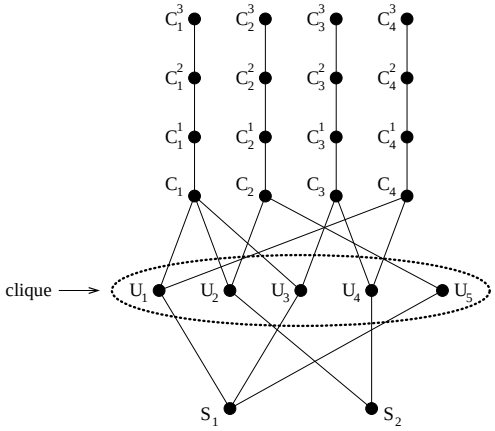

FIG. 3.3. *An example of H.*

Notice that in the above lemma, we didn't use the property that H is chordal. In fact, *any* square root would tell us how to do set splitting. In particular, *any chordal* root would tell us how to do set splitting. This completes the proof of NP-completeness of SQUARE OF CHORDAL GRAPH.

THEOREM 3.5. SQUARE OF CHORDAL GRAPH *is NP-complete.*

Since *any* square root would tell us how to do set splitting, we have the following results.

THEOREM 3.6. *For any class X of graphs which contains the class of chordal graphs,* SQUARE OF X GRAPH *is NP-complete.*

COROLLARY 3.7. *Given G, determine if there exist a weakly chordal graph H such that $H^2 = G$ is NP-complete.*

COROLLARY 3.8. *Given G, determine if there exist a perfect graph H such that $H^2 = G$ is NP-complete.*

3.2. Square roots of chordal graphs. In this subsection, we will show that given a chordal graph G, it is NP-complete to determine if there exists H such that $H^2 = G$. Notice that the proof is almost identical to that of the previous subsection; therefore, we will omit unnecessary details.

PROBLEM. SQUARE ROOT OF CHORDAL GRAPH
INSTANCE. A *chordal graph* $G = (V, E)$.
QUESTION. Does there exist a graph H such that $H^2 = G$?

The rest of this section shows that SQUARE ROOT OF CHORDAL GRAPH is NP-hard by reducing SET SPLITTING to it. It is clear that SQUARE ROOT OF CHORDAL GRAPH is in NP, since guessing the square root H and verifying that $H^2 = G$ can be easily done in polynomial time. Thus we will conclude that SQUARE ROOT OF CHORDAL GRAPH is NP-complete.

3.2.1. The reduction. Given an instance of SET SPLITTING, we construct an instance of SQUARE ROOT OF CHORDAL GRAPH. Let c_j be the set of elements in subset j, let $C = \{c_1, \ldots, c_m\}$, 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 3.1):

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.

Edges of G

- *Edges of tail vertices of subset vertices:* $\forall c_j \in C$
  $C_j^3 \leftrightarrow C_j^2, C_j^3 \leftrightarrow C_j^1$,
  $C_j^2 \leftrightarrow C_j^1, C_j^2 \leftrightarrow C_j$,
  $C_j^1 \leftrightarrow C_i$ for all i and $C_j^1 \leftrightarrow U_i$ for all $u_i \in c_j$.
- *Edges of subset vertices:* $\forall c_j \in C$
  $C_j \leftrightarrow S_1, C_j \leftrightarrow S_2, C_j \leftrightarrow U_i$ for all i and $C_j \leftrightarrow C_k$ for all k.
- *Edges of element vertices:* $\forall u_i \in U$
  $U_i \leftrightarrow U_j$ for $i \neq j$, $U_i \leftrightarrow S_1$ and $U_i \leftrightarrow S_2$.

LEMMA 3.9. *G is a chordal graph.*

Proof. We do this by showing a simplicial elimination ordering of G. For all C_j^3, they are simplicial and we eliminate them first. Then for any C_j^2, it is adjacent only to C_j^1 and C_j which are adjacent. So C_j^2 is simplicial for all j and we eliminate them. The union of the set of subset vertices and the set of element vertices induces a complete graph. For all C_j^1 and S_1 and S_2, their neighbor sets are a subset of that union. So all C_j^1 and S_1 and S_2 are simplicial and we can eliminate them. Finally a complete graph is left and this completes the proof that G is chordal. □

LEMMA 3.10. *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 exist a graph H such that $H^2 = G$.*

Proof. Edges of H.

- *Edges of subset vertices and its tail vertices:*
  $C_j^3 \leftrightarrow C_j^2, C_j^2 \leftrightarrow C_j^1, C_j^1 \leftrightarrow C_j$, $C_j \leftrightarrow U_i$ iff $u_i \in c_j$ and $C_j \leftrightarrow C_k$ for all k.
- *Edges of partition vertices:*
  $S_k \leftrightarrow U_i$ iff $u_i \in S_k$.

- *Edges of subset vertices:*
  $U_i \leftrightarrow U_j$ for $i \neq j$.

It is a routine matter to check that $H^2 = G$ (see [23] for a full proof). □

By observing that $\{C_j^3, C_j^2, C_j^1, C_j\}$ satisfies the properties of Lemma 3.1 and using the same argument as previous section, we can prove the following result.

LEMMA 3.11. *If H is a square root of G, 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.*

THEOREM 3.12. SQUARE ROOT OF CHORDAL GRAPH *is NP-complete.*

It should be pointed out that H is actually a Berge graph (i.e. there is no odd hole and no odd antihole in H); proofs are omitted. By the recent Strong Perfect Graph Theorem [4], it implies that finding a perfect graph square root of a chordal graph is also NP-complete.

THEOREM 3.13. *It is NP-complete to determine if a chordal graph is the square of a perfect graph.*

3.3. Squares of split graphs. In this subsection, we will show that to determine if G is the square of a split graph is NP-complete.

Recall that an undirected graph $G = (V, E)$ is defined to be *split* if there is a partition $V = S + C$ of its vertex set into a stable set S and a complete set C. There is no restriction on edges between vertices of S and vertices of C.

PROBLEM. SQUARE OF SPLIT GRAPH
INSTANCE. Graph $G = (V, E)$.
QUESTION. Does there exist a *split graph* H such that $G = H^2$?

The motivation of studying this problem is the similarity of the structure of split graphs and the structure of bipartite graphs. While the vertex set of a bipartite graph is partitioned into two independent set, the vertex set of a split graph is partitioned into a clique and an independent set. In [22], we prove that squares of bipartite graphs can be recognized in polynomial time. Note that since a split graph is of diameter at most 3, the tail structure can not be applied in the reduction. In fact, we use a totally different reduction for this problem.

Given an instance of INTERSECTION GRAPH BASIS, we transform it into an instance of SQUARE OF SPLIT GRAPH. Without loss of generality, we will assume that the graph in the instance of INTERSECTION GRAPH BASIS has no universal vertex and no isolated vertex. Recall that in the previous reduction, any square root will tell us the corresponding set splitting. In this reduction, we use the property that the square root is a split graph in order to find the corresponding intersection graph basis.

The transformation goes as follows. Given $G = (V, E)$, we construct a graph $G' = (V', E')$ by adding a set U of k universal vertices to G. Since G has no universal vertex, there are exactly k universal vertices in G'. We will show that G is an intersection graph of a family of sets whose union has cardinality at most k if and only if G' has a split root.

For example, given G as in Figure 3.4 and $k = 4$, Figure 3.4 shows the whole transformation process. In the figure, G is in the top left corner. G is the intersection graph for a family of sets whose union has cardinality 4. The 4-element set B is shown with $B[a] = \{1\}, B[b] = \{1, 3\}, B[c] = \{1, 2\}, B[d] = \{3, 4\}, B[e] = \{2, 4\}$ and $B[f] = \{4\}$. It is easy to verify that G is the intersection graph of B. The bottom left corner shows G' which comes from G by adding 4 universal vertices. Every vertex in G is adjacent to every vertex in U in G'. This is represented by the dotted lines between G and U in G'. Also, U itself is a clique in G'. The bottom right corner

shows H' which is a split square root of G'. The reader may verify that H' is a split square root of G'. Finally, we observe that there is a one-to-one correspondence between the solution to the INTERSECTION GRAPH BASIS problem and the solution to the SQUARE OF SPLIT GRAPH problem.

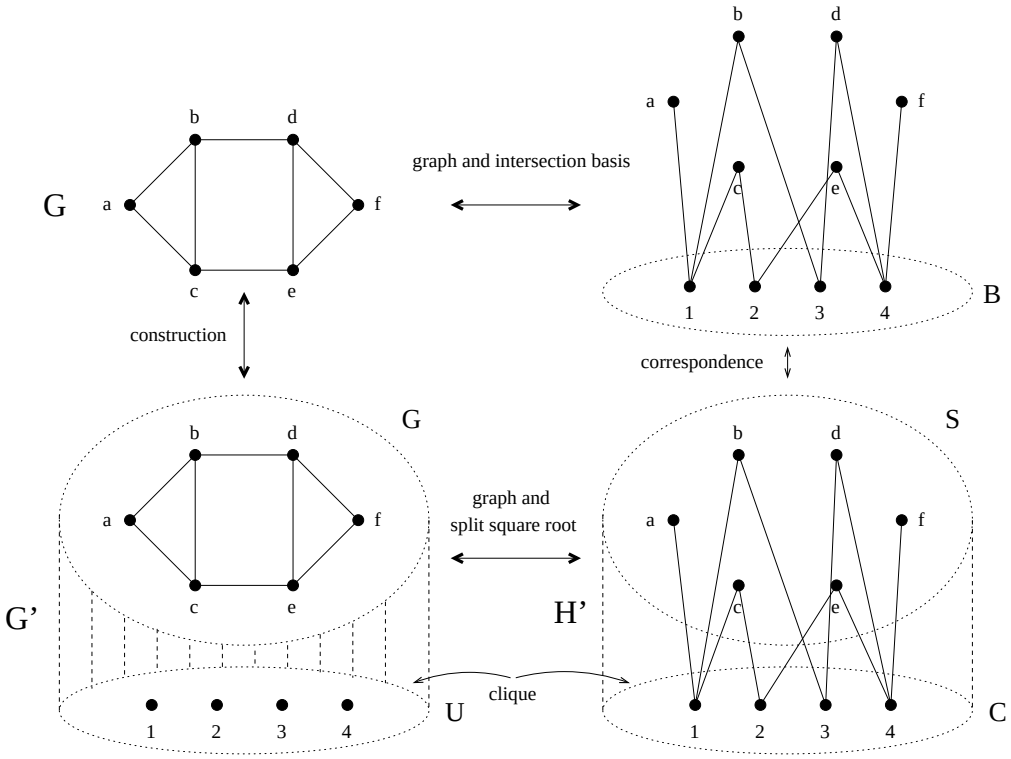

FIG. 3.4. *The whole transformation process of* SQUARE OF SPLIT GRAPH.

THEOREM 3.14. SQUARE OF SPLIT GRAPH *is NP-complete.*

Proof. We now argue that G is the intersection graph for a family of sets whose union has cardinality k or less if and only if there exist a split graph H such that $G' = H^2$.

First we prove the forward direction. If G is the intersection graph for a family of sets whose union has cardinality k or less, then we construct $H = (S + C, E)$ as follows. Each vertex c in C corresponds to an element in B. If $|B| < k$, we add some extra vertices to make $|C| = k$. Each vertex s in S corresponds to a vertex in G and s is adjacent to the vertices corresponding to $B[s]$. Now we check that $H^2 = G'$. Each vertex in the complete set is universal in H^2 and thus corresponds to a vertex in U. Two vertices u and v in the stable set have an edge in H^2 if and only if $B[u]$ and $B[v]$ are not disjoint. Thus $H^2[S]$ is precisely the G subgraph of G'. Hence $H^2 \cong G'$ as required.

Now we prove the reverse direction. If $H' = (S' + C', E')$ is a split square root of G', we construct the intersection graph basis as follows. Notice that a vertex in C' in H' is a universal vertex in H'^2. Since there are k vertices in G' that are universal vertices, there are at most k vertices in C' in H'. Furthermore, all the vertices in $G \subset V(G')$ must be in S' and so there is no edge between any two vertices in S'.

Two vertices u and v in the stable set have an edge in H'^2 if and only if $N_{H'}(u)$ and $N_{H'}(v)$ are not disjoint. Now we see that G has an intersection basis B such that $|B| \leq k$ by setting $B = C'$. For $u \in G$, $B[u] = N_{H'}(u)$ thereby showing that for any split root of G', we can construct an intersection graph basis of cardinality at most k. □

It is perhaps important to mention that, unlike the previous section, this result does not imply finding square roots of a more general class (e.g. chordal graphs) is NP-complete. It is because we use the property that the square root is a split graph to force all the vertices that are not universal in the square to form an independent set in the square root.

4. Concluding remarks. For k-TH POWER OF PROPER INTERVAL GRAPH, the complexity of our algorithm is exponential in k. It is open whether there is a polynomial time algorithm for k-TH POWER OF PROPER INTERVAL GRAPH when k is part of the input. Also, it is open whether there is a polynomial time algorithm to recognize squares of interval graphs, or more generally, k-th powers of interval graphs.

Acknowledgment. The authors wish to thank the National Science and Engineering Research Council of Canada for financial support. We also thank the anonymous referees for their helpful comments.

REFERENCES

[1] R. BALAKRISHNAN, P. PAULRAJA, *Powers of chordal graphs*, Australian Journal of Mathematics Series A, 35 (1983), pp. 211–217.
[2] A. BRANDSTÄDT, F. F. DRAGAN, V. D. CHEPOI, V. I. VOLOSHIN, *Dually chordal graphs*, SIAM J. Discrete Math., 11 (1998), pp. 437–455.
[3] J. M. CHANG, C. W. HO, M. T. KO, *Powers of Asteroidal Triple-free Graphs with Applications*, Ars Combinatoria, 67 (2003), pp. 161–173.
[4] M. CHUDNOVSKY, N. ROBERTSON, P. SEYMOUR, R. THOMAS, *The Strong Perfect Graph Theorem*, preprint, `http://www.math.gatech.edu/ thomas/spgc.html`.
[5] D. G. CORNEIL, *A simple 3-sweep LBFS algorithm for the recognition of unit interval graphs*, Discrete Applied Mathematics, to appear.
[6] P. DAMASCHKE, *Distances in cocomparability graphs and their powers*, Discrete Applied Mathematics, 35 (1992), pp. 67–72.
[7] X. DENG, P. HELL, J. HUANG, *Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs*, SIAM J. Comput., 25 (1996), pp. 390–403.
[8] P. DUCHET, *Classical perfect graphs*, Ann. Discrete Math., 21 (1984), pp. 67–96.
[9] F. ESCALANTE, L. MONTEJANO, T. ROJANO, *Characterization of n-path graphs and of graphs having nth root*, J. Combin. Theory B, 16 (1974), pp. 282–289.
[10] H. FLEISCHNER, *The square of every two-connected graph is Hamiltonian*, J. Combin. Theory B, 16 (1974), pp. 29–34.
[11] C. FLOTOW, *Graphs whose powers are chordal and graphs whose powers are interval graphs*, J. Graph Theory, 24 (1997), pp. 323–330.
[12] M. R. GAREY, D. S. JOHNSON, *Computers and Intractability - A Guide to the Theory of NP-completeness*, Freeman, Oxford, 1979.
[13] D. P. GELLER, *The square root of a digraph*, J. Combin. Theory B, 5 (1968), pp. 320–321.
[14] P. C. GILMORE, A. J. HOFFMAN, *A characterization of comparability graphs and of interval graphs*, Canad. J. Math., 16 (1964), pp. 539–548.
[15] F. HARARY, R. M. KARP, W. T. TUTTE, *A criterion for planarity of the square of a graph*, J. Combin. Theory, 2 (1967), pp. 395–405.
[16] F. HARARY, T. A. MCKEE, *The square of a chordal graph*, Discrete Mathematics, 128 (1994), pp. 165–172.
[17] P. HELL, R. SHAMIR, R. SHARAN, *A fully dynamic algorithm for recognizing and representing proper interval graphs*, SIAM J. Comput., 31 (2001), pp. 289–305.
[18] P. KEARNEY, D. CORNEIL, *Tree powers*, Journal of Algorithms, 29 (1998), pp. 111–131.
[19] L. T. KOU, L. J. STOCKMEYER, C. K. WONG, *Covering edges by cliques with regard to keyword conflicts and intersection graphs*, Comm. ACM, 21 (1978), pp. 135–138.

[20] R. Laskar, D. Shier, *On powers and centers of chordal graphs*, Discrete Applied Mathematics, 6 (1983), pp. 139–147.
[21] H. T. Lau, *Finding a Hamiltonian cycle in the square of a block*, PhD thesis, School of Computer Science, McGill University, Montreal, 1980.
[22] L. C. Lau, *Bipartite roots of graphs*, manuscript, 2003.
[23] L. C. Lau, *Roots of graphs*, Master thesis, University of Toronto, 2003.
[24] Guo-Hui Lin, Paul E. Kearney, Tao Jiang, *Phylogenetic k-Root and Steiner k-Root*, Lecture Notes in Computer Science, 1969 (2000), pp. 539–551.
[25] Y. L. Lin, S. Skiena, *Algorithms for square roots of graphs*, SIAM J. Disc. Math., 8 (1995), pp. 99–118.
[26] N. Linial, *Locality in distributed graph algorithms*, SIAM J. Comput., 21 (1992), pp. 193–201.
[27] A. Lubiw, *γ-free matrices*, Master's thesis, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, 1982.
[28] M. Molloy, M. R. Salavatipour, *Frequency Channel Assignment on Planar Networks*, Proceedings of 10th Annual European Symposium on Algorithms (ESA 2002), LNCS 2461, pp. 736–747.
[29] R. Motwani, M. Sudan, *Computing roots of graphs is hard*, Discrete Applied Mathematics, 54 (1994), pp. 81–88.
[30] A. Mukhopadhyay, *The square root of a graph*, J. Combin. Theory, 2 (1967), pp. 290–295.
[31] N. Nishimura, P. Ragde, D. Thilikos, *On graph powers for leaf-labeled trees*, Journal of Algorithms, 42 (2002), pp. 69–108.
[32] S, Ramachandran, *Planar graphs with square or cube root are four colorable*, J. Combin. Theory B, 24 (1978), pp. 362–364.
[33] A. Raychaudhuri, *On powers of interval and unit interval graphs*, Congr. Numer., 59 (1987), pp. 235–242.
[34] A. Raychaudhuri, *On powers of strongly chordal and circular arc graphs*, Ars Combin., 34 (1992), pp. 147–160.
[35] I. C. Ross, F. Harary, *The square of a tree*, Bell System Tech. J., 39 (1960), pp. 641–647.
[36] M. Sekanina, *On an ordering of the vertices of a graph*, Časopis Pěst. Math., 88 (1963), pp. 265–282.
[37] J. P. Spinrad, *Doubly lexical ordering of dense 0-1 matrices*, Inform. Process. Lett., 45 (1993), pp. 229–235.
[38] D. West, *Introduction to graph theory*, Prentice Hall, Ed. 2, 2001.