SIAM J. DISCRETE MATH. © 1998 Society for Industrial and Applied Mathematics
Vol. 11, No. 4, pp. 633–643, November 1998 008

k-NEIGHBORHOOD-COVERING AND -INDEPENDENCE PROBLEMS FOR CHORDAL GRAPHS[*]

SHIOW-FEN HWANG[†] AND GERARD J. CHANG[‡]

Abstract. Suppose $G = (V, E)$ is a simple graph and k is a fixed positive integer. A vertex z k-neighborhood-covers an edge (x, y) if $d(z, x) \leq k$ and $d(z, y) \leq k$. A k-neighborhood-covering set is a set C of vertices such that every edge in E is k-neighborhood-covered by some vertex in C. A k-neighborhood-independent set is a set of edges in which no two distinct edges can be k-neighborhood-covered by the same vertex in V. In this paper we first prove that the k-neighborhood-covering and the k-neighborhood-independence problems are NP-complete for chordal graphs. We then present a linear-time algorithm for finding a minimum k-neighborhood-covering set and a maximum k-neighborhood-independent set for a strongly chordal graph provided that a strong elimination ordering is given in advance.

Key words. k-neighborhood-covering, k-neighborhood-independence, chordal graph, strongly chordal graph, strong elimination order

AMS subject classifications. 05C80, 05C70, 68R10, 68Q20

PII. S0895480193244322

1. Introduction.

Domination is a natural model for location problems in operations research. This paper studies a variant of the domination problem we call k-neighborhood-covering. All graphs in this paper are simple, i.e., finite, undirected, loopless, and without multiple edges. In a graph $G = (V, E)$, the *length* of a path is the number of edges in the path. The *distance* $d(x, y)$ from vertex x to vertex y is the minimum length of a path from x to y; $d(x, y) = \infty$ if there is no path from x to y. A vertex x *k-dominates* another vertex y if $d(x, y) \leq k$. A vertex z *k-neighborhood-covers* an edge (x, y) if $d(z, x) \leq k$ and $d(z, y) \leq k$, or, equivalently, z k-dominates both x and y. A vertex set $C \subseteq V$ is a *k-neighborhood-covering set* if every edge in E is k-neighborhood-covered by some vertex in C. The *k-neighborhood-covering number* $\rho_N(G, k)$ of G is the minimum cardinality of a k-neighborhood-covering set. An edge set $I \subseteq E$ is a *k-neighborhood-independent set* if no two distinct edges in I are k-neighborhood-covered by the same vertex in V. The *k-neighborhood-independence number* $\alpha_N(G, k)$ of G is the maximum cardinality of a k-neighborhood-independent set.

For any graph G and any positive integer k, $\alpha_N(G, k)$ and $\rho_N(G, k)$ are related by the following obvious *max-min inequality*:

$$\alpha_N(G, k) \leq \rho_N(G, k).$$

For $k = 1$, Lehel and Tuza [11] proved that the above inequality is in fact an equality for odd-sun-free chordal graphs G and presented a linear-time algorithm for computing $\alpha_N(G, 1) = \rho_N(G, 1)$ for interval graphs. Wu [17] presented an $O(|V|^3)$-time algorithm for determining $\rho_N(G, 1) = \alpha_N(G, 1)$ for a strongly chordal graph G. Chang, Farber,

[*]Received by the editors February 16, 1993; accepted for publication (in revised form) September 6, 1997; published electronically September 1, 1998.

http://www.siam.org/journals/sidma/11-4/24432.html

[†]Department of Computer Science, Fen Chia University, Taichung 40724, Taiwan.

[‡]Department of Applied Mathematics, National Chiao Tung University, Hsinchu 30050, Taiwan (gjchang@math.nctu.edu.tw). The research of this author was supported in part by the National Science Council under grant NSC81-0208-M009-26 and in part by DIMACS.

and Tuza [6] presented linear-time algorithms for computing $\alpha_N(G,1) = \rho_N(G,1)$ for a strongly chordal graph G provided that a strong elimination order is known in advance. The purpose of this paper is to study $\alpha_N(G,k)$ and $\rho_N(G,k)$ for chordal graphs.

A graph is *chordal* (or *triangulated*) if every cycle of length greater than three has a chord, which is an edge joining two noncontiguous vertices in the cycle. In a graph $G = (V,E)$, the *neighborhood* $N(v)$ of a vertex v is the set of all vertices adjacent to v and the *closed neighborhood* $N[v] = N(v) \cup \{v\}$. A vertex v is *simplicial* if $N[v]$ is a clique. It is well known that a graph $G = (V,E)$ is chordal if and only if it has a *perfect elimination order*, i.e., an ordering $[v_1, v_2, \ldots, v_n]$ of V such that each v_i is a simplicial vertex of the subgraph G_i induced by $\{v_i, v_{i+1}, \ldots, v_n\}$ (see [8]).

An *s-sun* (or *incomplete trampoline of order s*) is a chordal graph having $2s$ vertices $a_1, a_2, \ldots, a_s, b_1, b_2, \ldots, b_s$ such that $(a_1, a_2, \ldots, a_s, a_1)$ is a cycle and each b_i has exactly two neighbors a_i and a_{i+1}, where $a_{n+1} = a_1$. A graph is *sun-free* (resp., *odd-sun-free*) if it contains no *s-sun* as a subgraph for all $s \geq 3$ (resp., for all odd $s \geq 3$) (see [4, 5]). Sun-free chordal graphs are called *strongly chordal graphs* in [7]. A vertex is *simple* if the set $\{N[u]\colon u \in N[v]\}$ can be linearly ordered by inclusion. Farber [7] proved that a graph $G = (V,E)$ is strongly chordal if and only if it has a *simple elimination order*, i.e., an ordering $[v_1, v_2, \ldots, v_n]$ of V such that each v_i is a simple vertex of G_i. A *strong elimination order* is a simple elimination order such that $N_{G_i}[v_j] \subseteq N_{G_i}[v_k]$ whenever $i \leq j \leq k$ and $v_j, v_k \in N_{G_i}[v_i]$. Anstee and Farber [2] presented an $O(|V|^3)$-time algorithm, Hoffman, Kolen, and Sakarovitch [9] presented an $O(|V|^3)$-time algorithm, Lubiw [12] presented an $O(L \log^2 L)$-time algorithm where $L = |V| + |E|$, Paige and Tarjan [13] presented an $O(L \log L)$-time algorithm, and Spinrad [16] presented an $O(|V|^2)$-time algorithm for finding a strong elimination order of a strongly chordal graph $G = (V,E)$.

The contents of this paper are as follows. In section 2, we prove that the k-neighborhood-covering and the k-neighborhood-independence problems are NP-complete for chordal graphs. Section 3 gives a linear-time algorithm for computing $\alpha_N(G,k)$ and $\rho_N(G,k)$ for a strongly chordal graph G provided that a strong elimination ordering is given in advance. Section 4 verifies the correctness of the algorithm. The algorithm in fact gives a k-neighborhood-covering set C^* and a k-neighborhood-independent set I^* with $|C^*| = |I^*|$. Consequently, C^* and I^* are optimal and $\alpha_N(G,k) = \rho_N(G,k)$ for any strongly chordal graph G.

In the rest of this section, we discuss the k-neighborhood-covering problem by means of the integer-linear programming method. The k-neighborhood-covering problem for a graph G is precisely the integer-linear programming problem

$$\text{(ILP)} \qquad \min \{1 \cdot x\colon\ Mx \geq 1 \text{ and } x \geq 0 \text{ integral}\},$$

where M is the 0-1 matrix whose rows are indexed by the edges of G and whose columns are indexed by the vertices and which possesses an entry of 1 for row e and column v if and only if v is within distance k of both ends of e. The k-neighborhood-independence problem is the following integer dual of (ILP):

$$\text{(ID)} \qquad \max \{y \cdot 1\colon\ yM \leq 1 \text{ and } y \geq 0 \text{ integral}\}.$$

We now claim that if G is strongly chordal, then matrix M is totally balanced. First, the 0-1 matrix A_k, with rows and columns both indexed by vertices of G and with a 1 for row u and column v if and only if u and v are within distance k, is totally balanced [12]. Second, the property of being totally balanced is preserved by the operation of adding a new row which is the intersection of two existing rows [1]. Thus for each edge (u, v) we can add a row which is the intersection of row u and row v—this new row has 1's for precisely the vertices x that are within distance k of both u and v. Finally, deleting the rows corresponding to vertices yields the matrix M.

Since M is totally balanced, the integrality conditions in (ILP) and (ID) can be dropped, thus giving a min-max equality and a polynomial-time algorithm by means of a greedy method [9]. This method also works for the weighted cases of the problems. The drawback of the method is that it takes more than linear-time to compute M. The main effort of this paper is to give a linear-time algorithm to solve these problems without explicit computation of M. This is similar to the case of solving the k-domination problem in strongly chordal graphs in linear-time without taking the k-power of the graph; see [3].

2. NP-completeness for chordal graphs.

In this section we show that for any fixed k, the k-neighborhood-covering and the k-neighborhood-independence problems are NP-complete for chordal graphs. We shall do this by reducing the problems with $k = 1$ for split graphs, which are known to be NP-complete in [6], to the problems for chordal graphs. A graph $G = (V, E)$ is *split* if its vertex set V is the disjoint union of a clique C and an independent set S. Split graphs are chordal.

THEOREM 2.1. *For any fixed positive integer k, the k-neighborhood-covering and the k-neighborhood-independence problems are NP-complete for chordal graphs.*

Proof. For any split graph $G = (V, E)$, whose vertex set V is the disjoint union of a clique C and an independent set S, construct the graph $G_k = (V_k, E_k)$ by attaching a path $s \equiv s_1, s_2, \ldots, s_k$ of length $k - 1$ to each vertex s in S. In other words,

$$V_k = C \cup \{s_i\colon\ s \in S \text{ and } 1 \le i \le k\} \quad\text{and}\quad E_k = E \cup \{(s_i, s_{i+1})\colon\ 1 \le i \le k-1\},$$

where s_1 is considered to be the same as s, G_k is clearly a chordal graph. We shall prove that $\rho_N(G_k, k) = \rho_N(G, 1)$ and $\alpha_N(G_k, k) = \alpha_N(G, 1)$. If these two equalities hold, then the theorem follows from the fact that the 1-neighborhood-covering and the 1-neighborhood-independence problems are NP-complete for split graphs (see [6]).

Suppose D is a minimum 1-neighborhood-covering set of G. Any edge in E is 1-neighborhood-covered and so k-neighborhood-covered by some vertex in D. For any edge $(s_i, s_{i+1}) \in E_k - E$, there exists some $x \in D$ such that $d(x, s) \le 1$. So $d(x, s_i) \le i < k$ and $d(x, s_{i+1}) \le i + 1 \le k$, i.e., x k-neighborhood-covers (s_i, s_{i+1}). Therefore D is a k-neighborhood-covering set of G_k. This gives $\rho_N(G_k, k) \le \rho_N(G, 1)$.

Conversely, suppose D is a minimum k-neighborhood-covering set of G_k. We can assume that $D \subseteq C$, otherwise any s_i in D can be replaced by a vertex adjacent to $s \equiv s_1$ in C to form a new minimum k-neighborhood-covering set. Consider any edge (x, y) in E. If x and y are both in C, then (x, y) is clearly 1-neighborhood-covered by any vertex in D. Suppose $x \in C$ and $y \in S$. Since D is a k-neighborhood-covering set of G_k, there exists some vertex z in D that k-neighborhood-covers (y_{k-1}, y_k). Both $z \in C$ and $d(z, y_k) \le k$ imply that z is adjacent to y and so z 1-neighborhood-covers (x, y). Therefore D is a 1-neighborhood-covering set of G. This gives $\rho_N(G_k, k) \ge \rho_N(G, 1)$. Together these inequalities imply that $\rho_N(G_k, k) = \rho_N(G, 1)$.

Suppose I is a minimum 1-neighborhood-independent set of G. It must be the case that each edge of I has one end in C and the other end in S. Also $S' = \{s \in S\colon$ there is some $(x, s) \in I\}$ is a 2-independent set; i.e., no two distinct vertices of S' have a common neighbor. So $I' = \{(s_{k-1}, s_k)\colon\ s \in S'\}$ is a k-neighborhood-independent set of G_k. This gives $\alpha_N(G_k, k) \ge \alpha_N(G, 1)$.

Conversely, suppose I' is a minimum k-neighborhood-independent set G_k. Each edge of I' must be of the form (s_{k-1}, s_k) for some $s \in S$. Also $S' = \{s \in S\colon \text{some } (s_{k-1}, s_k) \in I'\}$ is a 2-independent set. For any $s \in S'$, choose a neighbor s' of s in C. Then $I = \{(s', s)\colon s \in S'\}$ is a 1-neighborhood-independent set of G. This gives $\alpha_N(G_k, k) \le \alpha_N(G, 1)$. Together these inequalities imply that $\alpha_N(G_k, k) = \alpha_N(G, 1)$. ☐

3. The algorithm for strongly chordal graphs. In this section we set forth a linear-time algorithm for the k-neighborhood-covering and the k-neighborhood-independence problems for a strongly chordal graph G provided that a strong elimination ordering is given in advance. Without loss of generality, we may assume that G has no isolated vertices. The algorithm in fact gives a k-neighborhood-covering set C^* and a k-neighborhood-independent set I^* with $|C^*| = |I^*|$. By definitions and the max-min inequality,

$$|C^*| = |I^*| \le \alpha_N(G, k) \le \rho_N(G, k) \le |C^*|.$$

Hence all inequalities are equalities. Consequently, C^* and I^* are optimal and $\alpha_N(G, k) = \rho_N(G, k)$ for any strongly chordal graph G.

Suppose $G = (V, E)$ is a strongly chordal graph for which a strong elimination order $[v_1, v_2, \ldots, v_n]$ is given. Note that this is also a perfect elimination order of the chordal graph G. For simplicity we identify vertex v_i as i, and hence $[1, 2, \ldots, n]$ is a strong elimination order. For any vertices i and j, $i \le j$, let

$$N_i(j) = \{l \in\colon (j, l) \in E \text{ and } l \ge i\} \text{ and}$$
$$N_i[j] = N_i(j) \cup \{j\}.$$

As in [7], we use $N^+(i)$ for $N_i(i)$ and $N^+[i]$ for $N_i[i]$. A useful property of a chordal graph G, in which $[1, 2, \ldots, n]$ is a perfect elimination order, is that

$$N^+[i] \text{ is a clique for any } i. \tag{3.1}$$

A useful property of a strongly chordal graph G, in which $[1, 2, \ldots, n]$ is a strong elimination order, is that

$$j, l \in N^+[i] \text{ and } i \le j \le l \text{ imply } N_i[j] \subseteq N_i[l]. \tag{3.2}$$

(3.2) states that in the graph G_i induced by $\{i, i+1, \ldots, n\}$, the maximum neighbor of i has the largest closed neighborhood among all neighbors of i. So, the maximum neighbor is a most powerful dominating vertex. This is important to the development that follows. (3.1) and (3.2) are also used frequently in the proofs of lemmas and theorems in section 4.

The idea behind our algorithm for the k-neighborhood-covering and the k-neighborhood-independence problems is analogous to, but much more complicated than, that behind the algorithms for the k-domination problem (see [3, 15]). In fact, a $(k-1)$-dominating set is a k-neighborhood-covering set. However, the converse is not true. Note that a vertex set C is a k-neighborhood-covering set if and only if for any edge e in G, either one end vertex of e is $(k-1)$-dominated by some vertex in C, or both end vertices of e are exactly k-distant from the same vertex in C. So our algorithm retains the spirit of the $(k-1)$-domination problem with special attention to cases in which a *critical edge* e occurs; i.e., both end vertices of e are exactly k-distant from the same vertex in C. To handle critical edges, we employ some of the ideas in [6] for 1-neighborhood-covering and -independence.

The algorithm processes vertices in the order $1, 2, \ldots, n$. Initially the k-neighborhood-covering set C and the k-neighborhood-independent set I are empty. In iteration i, the algorithm determines whether vertex i must be put into C. If the answer is positive, the algorithm also finds an edge to put into I. After processing, vertex i is deleted from the graph and $i+1$ becomes a simple vertex of the remaining graph.

For technical reasons, we associate each vertex i with two nonnegative integers $a(i)$ and $b(i)$, two vertices $A(i)$ and $s(i)$, and a subset $N_0^+(i)$ of $N^+(i)$. The meanings of these items are as follows. For each vertex i the algorithm must eventually include some vertex that is within distance $a(i)$ from i in the k-neighborhood-covering set C. At any time, there is a vertex in the current C that is within distance $b(i)$ from i. Both $a(i)$ and $b(i)$ keep decreasing as the algorithm proceeds. $a(i)$ decreases when i is the maximum neighbor of a smaller vertex i' that is not properly neighborhood-covered by a vertex of C within distance $a(i')$ in iteration i'. In this case, $a(i)$ is set to $a(i')-1$. Similarly, in a previous iteration, $a(i')$ was set to $a(i'')-1$. Continuing this argument, there exists a smallest vertex i^* that forces $\ldots, i'', i', i$ to decrease their $a(\cdot)$ values, although $a(i^*)$ never changes. We use $A(i)$ to denote this *initial vertex* i^* from which $a(i)$ decreases. $s(i)$ is the optimal candidate for j such that (i,j) is put into I. $N_0^+(i)$ is a set of candidates for $s(i)$. More precisely, $N_0^+(i)$ contains those $j \in N^+(i)$ such that there is no vertex i' 1-neighborhood-covering (i,j) and i' itself is $(k-1)$-dominated by a vertex in the current C. $s(i)$ is chosen to be min $N_0^+(i)$ in iteration i. If in iteration i the algorithm determines that i should be put into C, it will also put the edge $(A(i), s(A(i)))$ into I. Initially, $a(i)=k$, $b(i)=\infty$, $A(i)=i$, $s(i)=\infty$, and $N_0^+(i)=N^+(i)$ for all $i \in V$. The algorithm processes vertex i according to one of the following cases.

When $a(i)=0$, vertex i must be put into C. When $a(i)<b(i)$ and $N^+(i)=\emptyset$, all vertices in the current C are farther than distance $a(i)$ from i and no vertex $j>i$ is adjacent to i. So, we need to put i into C. In these two cases, we also put the edge $(A(i), s(A(i)))$ into I. Since i is now in C, $b(i)$ is set to 0.

Suppose $a(j) \geq a(i)$ and $b(j) \geq a(i)$ for all $j \in N^+(i)$. When $0<a(i)<b(i)$ and $N^+(i) \neq \emptyset$, all vertices in the current C are farther than distance $a(i)$ from i. In this case, we need to find a vertex no farther than $a(i)-1$ from the maximum neighbor m of i, since for any vertex $j>i$ there is a shortest path from j to i that passes through m. When $a(i)=b(i)=k$ and $N_0^+(i) \neq \emptyset$, there is one vertex in the current C that is k-distant from vertex i. However, this vertex does not k-neighborhood-cover any edge (i,j) with $j \in N_0^+(i)$. Again, we need to find a vertex that $(k-1)$-dominates m and so k-neighborhood-covers all edges (i,j) with $j \in N_0^+(i)$.

When $a(i) \geq b(i)$ and $k > b(i)$, there is a vertex in the current C that is within distance $a(i)$ from i. This vertex in fact $(k-1)$-dominates i and so k-neighborhood-covers all edges incident to i. When $a(i)=b(i)=k$ and $N_0^+(i)=\emptyset$, by the definition of $N_0^+(i)$, any edge (i,j) with $j \in N^+(i)$ is k-neighborhood-covered by some vertex in C. When $a(j)<a(i)$ or $b(j)<a(i)$ for some $j \in N^+(i)$, a vertex of C that is within distance $a(j)$ from j must be within distance $a(i)$ from i. In these three cases, we do not need to do anything.

Finally, we need to update $b(j)$ and $N_0^+(j)$ for all vertices $j \in N^+(i)$.

We can summarize the procedure described above in the following algorithm:

Algorithm CI.

Input. A strongly chordal graph $G=(V,E)$ without isolated vertices and in which

$[1, 2, \dots, n]$ is a strong elimination order.

Output. A minimum k-neighborhood-covering set C and a maximum k-neighborhood-independent set I of G.

Method.

```
 1.   $C \longleftarrow \emptyset$;  $I \longleftarrow \emptyset$;
 2.   **for all** $i \in V$ **do**
 3.       $[a(i) \longleftarrow k$;  $b(i) \longleftarrow \infty$;  $A(i) \longleftarrow i$;  $s(i) \longleftarrow \infty$;  $N_0^+(i) \longleftarrow N^+(i)]$;
 4.   **for** $i = 1$ **to** $n$ **do**
 5.       **Case 1**: $a(i) = 0$ **or** $(a(i) < b(i)$ **and** $N^+(i) = \emptyset)$
 6.           $C \longleftarrow C \cup \{i\}$;
 7.           $I \longleftarrow I \cup \{(A(i), s(A(i)))\}$;
 8.           $b(i) \longleftarrow 0$;
 9.       **Case 2:** $((0 < a(i) < b(i)$ **and** $N^+(i) \neq \emptyset)$ **or** $(a(i) = b(i) = k$
                $N_0^+(i) \neq \emptyset))$ **and** $(a(j) \geq a(i)$ **and** $b(j) \geq a(i)$ **for all** $j \in N^+(i))$
11.           $m \longleftarrow \max N^+(i)$;
12.           $a(m) \longleftarrow a(i) - 1$;
13.           $A(m) \longleftarrow A(i)$;
14.           $s(i) \longleftarrow \min N_0^+(i)$;     $\{$where $\min \emptyset = \infty\}$
15.       **Case 3:** $(a(i) \geq b(i)$ **and** $k > b(i))$ **or** $(a(i) = b(i) = k$ **and** $N_0^+(i) = \emptyset)$
16.                 **or** $(a(j) < a(i)$ **or** $b(j) < a(i)$ **for all** $j \in N^+(i))$
17.           do nothing;
18.       **for all** $j \in N^+(i)$ **do** $b(j) \longleftarrow \min\{b(j), b(i) + 1\}$;
19.       $R \longleftarrow \{j \in N_0^+(i) : a(j) = b(j) = b(i) + 1 = k\}$;
20.       **for all** $j \in R$ **do** $N_0^+(j) \longleftarrow N_0^+(j) - R$;
21.       **end for.**
```

4. Correctness of the algorithm.

This section proves the correctness of Algorithm CI. First, we note that during the execution of the algorithm, $a(i)$, $b(i)$, $A(i)$, $s(i)$, C, and I are updated. We shall denote their final values by $a^*(i)$, $b^*(i)$, $A^*(i)$, $s^*(i)$, C^*, and I^*, respectively. Note that $a(i)$, $b(i)$, and $A(i)$ keep decreasing and stay at their final values from the beginning of iteration i. $s(i)$ only changes from ∞ to $\min N_0^+(i)$ in iteration i when Case 2 of the algorithm holds.

Our proof of the correctness of Algorithm CI is based on proving that C^* is a k-neighborhood-covering set (Theorem 4.4), I^* is a k-neighborhood-independent set (Theorem 4.11), and $|C^*| = |I^*|$ (Theorem 4.7). If these conditions hold, C^* is a minimum k-neighborhood-covering set, I^* is a maximum k-neighborhood-independent set, and $\alpha_N(G, k) = \rho_N(G, k)$.

LEMMA 4.1. *For any vertex $x \in V$, there exists some $y \in C^*$ such that $d(x, y) \leq b^*(x)$.*

Proof. We claim that in any iteration, for any vertex $x \in V$, there exists some $y \in C$ such that $d(x, y) \leq b(x)$. Initially, $b(x) = \infty$ and $C = \emptyset$. The claim holds if we interpret it to be $\min_{y \in C} d(x, y) \leq b(x)$ and $\min \emptyset = \infty$. $b(x)$ only changes its value in lines 8 and 18. If $b(x)$ is reset to 0 in line 8, then x is added to C in line 6. So $d(x, x) = 0 = b(x)$ for $x \in C$. Suppose $b(x)$ is reset to $b(i) + 1$ in line 18 for $x \in N^+(i)$ and $b(x) > b(i) + 1$; it then follows from the induction hypothesis that there must be some $y \in C$ such that $d(i, y) \leq b(i)$. Therefore, $d(x, y) \leq d(x, i) + d(i, y) \leq 1 + b(i) < b(x)$. This proves the claim. Consequently, the lemma holds. ▯

LEMMA 4.2. *For any vertex $x \in V$, one of the following three statements holds.*

(1) $b^*(x) \leq a^*(x)$.

(2) $b^*(j) < a^*(x)$ *for some* $j \in N^+(x)$.

(3) *$a^*(j) < a^*(x)$ for some $j \in N^+(x)$.*

Proof. We shall prove the lemma by considering the following cases in iteration x.

If Case 1 of the algorithm holds, then $b^*(x) = 0 \le a^*(x)$; i.e., (1) holds.

If Case 2 of the algorithm holds, then there exists $j = \max N^+(x)$ such that $a(j) = a(x) - 1$ and so $a^*(j) \le a(j) < a(x) = a^*(x)$; i.e., (3) holds.

If Case 3 of the algorithm holds, then $b(x) \le a(x)$ or there exists $j \in N^+(x)$ such that $a(j) < a(x)$ or $b(j) < a(x)$. For the first case, $b^*(x) \le b(x) \le a(x) = a^*(x)$; i.e., (1) holds. For the second case, $a^*(j) \le a(j) < a(x) = a^*(x)$; i.e., (3) holds. For the third case, $b^*(j) \le b(j) < a(x) = a^*(x)$; i.e., (2) holds. □

LEMMA 4.3. *For any vertex $x \in V$, there exists some $y \in C^*$ such that $d(x, y) \le a^*(x)$.*

Proof. Repeatedly apply Lemma 4.2 to get a sequence $x \equiv x_0, x_1, \dots, x_{r-1}, x_r$ such that $x_i \in N^+(x_{i-1})$ for $1 \le i \le r$ and

$$b^*(x_r) \le a^*(x_r) < a^*(x_{r-1}) < \cdots < a^*(x_1) < a^*(x_0) = a^*(x)$$

$$\text{or} \qquad b^*(x_r) < a^*(x_{r-1}) < \cdots < a^*(x_1) < a^*(x_0) = a^*(x).$$

Then $r + b^*(x_r) \le a^*(x)$. By Lemma 4.1, there exists some $y \in C^*$ such that $d(x_r, y) \le b^*(x_r)$. Hence $d(x, y) \le d(x, x_r) + d(x_r, y) \le r + b^*(x_r) \le a^*(x)$. □

THEOREM 4.4. *C^* is a k-neighborhood-covering set of G.*

Proof. Suppose (x, z) is an edge with $x < z$, i.e., $z \in N^+(x)$. We shall prove the theorem according to (1), (2), and (3) of Lemma 4.2.

(1) $b^*(x) \le a^*(x)$. By Lemma 4.1, there exists some $y \in C^*$ such that $d(x, y) \le b^*(x) \le a^*(x) \le k$. If $d(x, y) \le k - 1$, then y k-neighborhood-covers (x, z). So we may assume that $d(x, y) = b^*(x) = a^*(x) = k$. Since Lemma 4.2 (1) holds only when Case 2 of the algorithm does not, $N_0^+(x) = \emptyset$ in iteration x. By $z \in N^+(x)$ and the updating rule for $N_0^+(j)$ in lines 18 and 19, there exists an i such that $x, z \in N_0^+(i)$ and $b(i) = k - 1$ in iteration i. By Lemma 4.1, there exists some $y \in C^*$ such that $d(i, y) \le b(i) = k - 1$. Hence y $(k-1)$-dominates i and i 1-neighborhood-covers (x, z); i.e., y k-neighborhood-covers (x, z).

(2) $b^*(j) < a^*(x)$ for some $j \in N^+(x)$. By Lemma 4.1, there exists some $y \in C^*$ such that $d(j, y) \le b^*(y) < a^*(x) \le k$; i.e., y $(k-1)$-dominates j. Since $z, j \in N^+(x)$, j 1-neighborhood-covers (x, z) by (3.1). So y k-neighborhood-covers (x, z).

(3) $a^*(j) < a^*(x)$ for some $j \in N^+(x)$. By Lemma 4.3, there exists some $y \in C^*$ such that $d(j, y) \le a^*(y) < a^*(x) \le k$; i.e., y $(k-1)$-dominates j. Since $z, j \in N^+(x)$, j 1-neighborhood-covers (x, z) by (3.1). So y k-neighborhood-covers (x, z). □

LEMMA 4.5. *If $y \in C^*$, then $a^*(y) < k$.*

Proof. Since $y \in C^*$, Case 1 of the algorithm holds in iteration y, i.e., $a(y) = 0$ or $a(y) < b(y)$ with $N^+(y) = \emptyset$. For the former case, clearly, $a^*(y) < k$. For the latter case, since G has no isolated vertex, there exists a largest neighbor w of y. Note that $y \in N^+(w)$. Suppose $N^+(w)$ contains a vertex y' other than y. Then by (3.1) y' is adjacent to y. So y' is a neighbor of y that is larger than w, which is a contradiction. Thus $N^+(w) = \{y\}$ and $y = \max N^+(w)$. We now consider the following cases in iteration w.

If Case 1 of the algorithm holds, then by line 8 $b(w) = 0$ and so, by line 18, $b(y) \le 1$. Thus $b(y) \le 1$ in iteration y and so $a^*(y) \le a(y) < b(y) \le 1 \le k$.

If Case 2 of the algorithm holds, then $a^*(y) \le a(y) = a(w) - 1 < k$.

If Case 3 of the algorithm holds, then $b(w) < k$ or $a(w) = b(w) = k$ with $N_0^+(w) = \emptyset$ or $a(y) < a(w)$ or $b(y) < a(w)$. For the first case, $b(y) \le k$ by line 18.

$b(y) \leq k$ still holds in iteration y and so $a^*(y) \leq a(y) < b(y) \leq k$. For the second case, by the updating rule in lines 19 and 20, there exists some i such that $w, y \in N_0^+(i)$ and $a(y) = b(y) = k$ in iteration i. Thus $b(y) \leq k$ still holds in iteration y and so $a^*(y) \leq a(y) < b(y) \leq k$. For the third case, $a^*(y) \leq a(y) < a(w) \leq k$. For the fourth case, $b(y) < a(w)$ still holds in iteration y and so $a^*(y) \leq a(y) < b(y) < a(w) \leq k$. ∎

LEMMA 4.6. *For any $\bar{x} \in C^*$ with $A(\bar{x}) = x$, there exists a unique increasing path $x \equiv \bar{x}_0, \bar{x}_1, \ldots, \bar{x}_u \equiv \bar{x}$ from x to $\bar{x}$ (with $u \geq 1$) such that $\bar{x}_i = \max N^+(\bar{x}_{i-1})$ for $1 \leq i \leq u$ and $A^*(\bar{x}_i) = x$ and $a^*(\bar{x}_i) = k - i$ for $0 \leq i \leq u$. Furthermore, $s^*(x) \neq \infty$ and $A^*(y) \neq A^*(\bar{x})$ for any $y \in C^* - \{\bar{x}\}$. Also, $u = k$ when $N^+(\bar{x}) \neq \emptyset$.*

Proof. Let $u = k - a^*(\bar{x})$ and $\bar{x}_u = \bar{x}$, i.e., $a^*(\bar{x}_u) = k - u$. By Lemma 4.5, $u \geq 1$. Note that $a^*(\bar{x}_u)$ is initially k and decreases only when Case 2 of the algorithm holds in some iteration i and $\bar{x}_u = \max N^+(i)$. There may be several such i. Let $\bar{x}_{u-1}$ be the maximum of all such i. In this case, $\bar{x}_u = \max N^+(\bar{x}_{u-1})$, $A^*(\bar{x}_u) = A^*(\bar{x}_{u-1})$, and $a^*(\bar{x}_{u-1}) = a^*(\bar{x}_u) + 1 = k - (u-1)$. Continuing the same argument, we get an increasing path $x \equiv \bar{x}_0, \bar{x}_1, \ldots, \bar{x}_u \equiv \bar{x}$ such that $\bar{x}_i = \max N^+(\bar{x}_{i-1})$ for $1 \leq i \leq u$, $A^*(\bar{x}_0) = A^*(\bar{x}_1) = \ldots = A^*(\bar{x}_u)$, and $a^*(\bar{x}_i) = k - i$ for $0 \leq i \leq k$. Since $a^*(\bar{x}_0) = k$, $A^*(\bar{x}_0)$ keeps its original value x, i.e., $A^*(\bar{x}_i) = x$ for all $0 \leq i \leq u$.

Furthermore, since $\bar{x}_i$ is uniquely determined by $\bar{x}_{i-1}$, such a path is unique and there is no $y \in C^* - \{\bar{x}\}$ with $A^*(y) = A^*(\bar{x})$.

Since $a^*(x) = k$, $a(x) = k$ at any time. In iteration x, Case 2 holds, i.e., $0 < a(x) < b(x)$ with $N^+(x) \neq \emptyset$ or $a(x) = b(x) = k$ with $N_0^+(x) \neq \emptyset$. For the latter case, $s^*(x) \neq \infty$ by line 14. For the former case, $b(x) > k$ in iteration x and so $b(x) > k$ in any previous iteration. By the definition of R in line 18, $x \notin R$ in any previous iteration. Hence by line 19, $N_0^+(x) = N^+(x) \neq \emptyset$ in iteration x. Again, $s^*(x) \neq \infty$.

If $N^+(\bar{x}) \neq \emptyset$, then $a(\bar{x}) = 0$ in line 5 in iteration $\bar{x}$. So, $k - u = a^*(\bar{x}_u) = a^*(\bar{x}) = 0$, i.e., $u = k$. ∎

We shall call the unique path $x \equiv \bar{x}_0, \bar{x}_1, \ldots, \bar{x}_u \equiv \bar{x}$ from x to $\bar{x}$ (with $u \geq 1$) in Lemma 4.6 the *maximum path* for $\bar{x}$. For any $\bar{x} \in C^*$, the corresponding edge $(x, x') \in I^*$, where $x = A^*(\bar{x})$ and $x' = s^*(x)$. Note that Case 2 holds in iteration $\bar{x}_i$ for $0 \leq i \leq u-1$ and Case 1 holds in iteration $\bar{x}_u \equiv \bar{x}$. Also, in Lemma 4.6, instead of saying $a^*(\bar{x}_i) = k - i$, we can say $a(\bar{x}_i) = k - i$ in iteration $\bar{x}_{i-1}$ when line 12 is executed for $1 \leq i \leq u$. $\bar{x}_1, x' \in N^+(x)$ imply that $d(\bar{x}_1, x') \leq 1$ by (3.1) and so $\bar{x}_1$ 1-neighborhood-covers (x, x'). This together with $d(\bar{x}_1, \bar{x}_p) \leq p - 1$ implies that $\bar{x}_p$ p-neighborhood-covers (x, x') for $1 \leq p \leq u$.

THEOREM 4.7. $|C^*| = |I^*|$.

Proof. By Lemma 4.6, for any two distinct vertices y, $y' \in C^*$, $A^*(y) \neq A^*(y')$, $s(A^*(y)) \neq \infty$, and $s(A^*(y')) \neq \infty$. Hence the theorem holds, because each time a new vertex is added to C in line 6 a new edge is added to I in line 7. ∎

LEMMA 4.8. *If vertex z k-neighborhood-covers an edge (x, x'), then there exists a shortest x-z path $x \equiv x_0, x_1, \ldots, x_r \equiv z$ such that $r \leq k$ and each x_p $(k-1)$-neighborhood-covers (x, x') for $1 \leq p \leq r - 1$.*

Proof. The lemma is easy when $d(x, z) = 1 + d(x', z)$ or $d(x', z) = 1 + d(x, z)$. For the case in which $d(x, z) = d(x', z) \leq k$, the lemma follows from the fact that there exists a vertex w adjacent to both x and x' such that $d(w, z) = d(x, z) - 1$. (See [10, Lemma 1 (d)].) ∎

LEMMA 4.9. *Any shortest x_0-x_r path $x_0, x_1, \ldots, x_r$ is unimodal, i.e., $x_0 < x_1 < \cdots < x_{i-1} < x_i$ and $x_i > x_{i+1} > \cdots > x_r$ for some $0 \leq i \leq r$.*

Proof. If the sequence is not unimodal, then $x_{i-1}, x_{i+1} \in N^+(x_i)$ for some i with $1 \leq i \leq r - 1$. By Property (3.1), $x_{i-1}x_{i-1} \in E$ and so $x_0, x_1, \ldots, x_r$ is not a shortest path. ∎

Lemma 4.10. *If $y_0,\ y_1,\ldots,y_r$ is an increasing path and $y_0 \in C^*$, then $b(y_i) \le i$ after iteration y_{i-1} for $1 \le i \le r$. Consequently, $b(y_i) \le i$ in and after iteration y_i for $0 \le i \le r$.*

Proof. We shall prove the lemma by induction on i. If $i = 1$, then in iteration y_0, $b(y_0) = 0$ by line 8 of the algorithm and so $b(y_1) \le 1$ by line 18. Suppose $i \ge 2$ and $b(y_{i-1}) \le i-1$ after iteration y_{i-2}. In iteration y_{i-1}, by line 18, we have $b(y_i) \le b(y_{i-1}) + 1 \le i$. Hence the lemma holds. ∎

Theorem 4.11. *I^* is a k-neighborhood-independent set of G.*

Proof. Suppose I^* is not k-neighborhood-independent; i.e., there exists some vertex $z \in V$ that k-neighborhood-covers two distinct edges (x, x') and (y, y') in I^*, where $x' = s^*(x)$ and $y' = s^*(y)$. Without loss of generality, we may assume that z is set as large as possible.

By the algorithm, there exist $\bar{x}, \bar{y} \in C^*$ such that $x = A^*(\bar{x})$ and $y = A^*(\bar{y})$. Let $x \equiv \bar{x}_0,\ \bar{x}_1,\ldots,\bar{x}_u \equiv \bar{x}$ be the maximum path for $\bar{x}$ and $y \equiv \bar{y}_0,\ \bar{y}_1,\ldots,\bar{y}_v \equiv \bar{y}$ the maximum path for $\bar{y}$. Since $x \ne y$, $\bar{x} \ne \bar{y}$ by Lemma 4.6. Also, each $\bar{x}_p$ p-neighborhood-covers (x, x') for $1 \le p \le u$ and each $\bar{y}_q$ q-neighborhood-covers (y, y') for $1 \le q \le v$.

By Lemma 4.8, there exists a shortest x-z path P: $x \equiv x_0,\ x_1,\ldots,x_r \equiv z$ with $r \le k$ and a shortest y-z path $y \equiv y_0,\ y_1,\ldots,y_t \equiv z$ with $t \le k$ such that each x_p $(k-1)$-neighborhood-covers (x, x') for $1 \le p \le r-1$ and each y_q $(k-1)$-neighborhood-covers (y, y') for $1 \le q \le t-1$. By Lemma 4.9, $x_0 < x_1 < \cdots < x_i$ and $x_i > x_{i+1} > \cdots > x_r$ for some $0 \le i \le r$ and $y_0 < y_1 < \cdots < y_j$ and $y_j > y_{j+1} > \cdots > y_t$ for some $0 \le j \le t$.

Let i^* be the largest index such that $x_p = \bar{x}_p$ for $0 \le p \le i^*$. We may assume that i^* is as large as possible. Note that $i^* \le i$. For the case of $i^*+1 \le i$, $i^*+1 \le u$, otherwise $u < i^*+1$ would imply that $u < i \le r \le k$, but $x^*_{i+1} \in N^+(\bar{x}_u) = N^+(\bar{x})$ contradicts the last statement of Lemma 4.6. Since $x_{i^*+1}\bar{x}_{i^*+1} \in N^+(x_{i^*})$, by (3.1) $x_{i^*+1}\bar{x}_{i^*+1} \in E$. By the fact that $\bar{x}_{i^*+1} = \max N^+(\bar{x}_{i^*})$, we have $x_{i^*+1} < \bar{x}_{i_*+1}$. Now $i^*+1 = i$, otherwise $i^*+2 \le i$ would imply that $\bar{x}_{i^*+1}, x_{i^*+2} \in N^+(x_{i^*+1})$, which in turn implies $\bar{x}_{i^*+1}x_{i^*+2} \in E$ and so P': $\bar{x}_0,\ldots,\bar{x}_{i^*},\ x_{i^*+1},\ x_{i^*+2},\ x_{i^*+3}\ldots,x_r$ is a path with $i^*(P') > i^*(P)$. In conclusion, either $x_p = \bar{x}_p$ for $0 \le p \le i$ (when $i = i^*$) or $x_p = \bar{x}_p$ for $0 \le p \le i-1$ with $\bar{x}_{i-1} < x_i < \bar{x}_i$ form a clique of 3 vertices (when $i = i^*+1$). Similarly, we may assume that either $y_q = \bar{y}_q$ for $0 \le q \le j$ (when $j = j^*$) or $y_q = \bar{y}_q$ for $0 \le q \le j-1$ with $\bar{y}_{j-1} < y_j < \bar{y}_j$ form a clique of 3 vertices (when $j = j^*+1$).

We now claim that $z \ne \bar{y}$. Suppose, to the contrary, $z = \bar{y} \in C^*$. Suppose $i^* = i = r$. Since $\bar{x}_u = \bar{x} \ne \bar{y} = z = x_r = \bar{x}_{r^*}$, $r^* \le u-1$. But $\bar{x}_{r^*} \in C$ implies that Case 1 holds in iteration $\bar{x}_{i^*}$, in contradiction to the fact that Case 2 holds in iteration $\bar{x}_p$ for $0 \le p \le u-1$. Therefore, either $i^*+1 \le i = r$ or $i < r$. In any case, $r > 0$. Suppose $i = 0$. Consider the increasing path $z \equiv x_r,\ x_{r-1},\ldots,x_1,\ x_0 \equiv x \equiv \bar{x}_0$. By Lemma 4.10, $b(x_1) \le r-1$ in iteration x_1 and so, by line 18, $b(x_0) \le b(x_1)+1 \le r$ in iteration x_1. Then $b(x) \le r \le k$ in iteration x. On the other hand, by Lemma 4.6, $a^*(x) = a(\bar{x}_0) = k$ and so, $a(x) = k$ in any iteration. Then $a(x) = b(x) = k$ in iteration x, since Case 2 of the algorithm holds. Thus, $b(x_1) = k-1$ and $b(x) = k$ in iteration x_1, and $a(x) = b(x) = k = r$ in iteration x. By line 19, $x \in R$ in iteration x_1. Now, $d(z, x) = r = k$. Since $x' > x$, we also have $d(z, x') = k$. By Lemma 4.8, we may assume that x' is adjacent to x_1. By line 18, $b(x') \le b(x_1)+1 \le k$ in iteration x_1

and hence $b(x') \le k$ in iteration x. Since Case 2 holds in iteration x, $a(x') \ge a(x) = k$ and $b(x') \ge a(x) = k$. So, $a(x') = b(x') = k$ in iterations x and x_1. By line 19, $x' \in R$ in iteration x_1. Therefore, by line 20, $x' \notin N_0^+(x)$ after iteration x_1 is completed. In iteration x, it is impossible that $x' = s(x) \equiv N_0^+(x)$. Thus $i \ge 1$.

First, consider the case of $i^* + 1 \le i = r$, i.e., $x_i = z = \bar{y} = \bar{y}_v \in C^*$. If $\bar{x}_{i-1} \le \bar{y}_{v-1}$, then $\bar{y}_{v-1} \in N_{\bar{x}_{i-1}}[x_i]$. By (3.2), $N_{\bar{x}_{i-1}}[x_i] \subseteq N_{\bar{x}_{i-1}}[\bar{x}_i]$. Thus, $\bar{y}_{v-1}$ is adjacent to $\bar{x}_i > x_i = \bar{y}_v$, in contradiction to $\bar{y}_v = \max N^+[\bar{y}_{v-1}]$. So, $\bar{y}_{v-1} < \bar{x}_{i-1}$. In iteration $\bar{y}_{v-1}$, $a(\bar{y}_v) = k - v$ by line 12 of the algorithm and Lemma 4.6. Also, $N^+(\bar{y}_v) \ne \emptyset$ since it contains $\bar{x}_i$. By Lemma 4.6, $v = k$. Then, $a(\bar{y}_v) = 0$ in and after iteration $\bar{y}_{v-1}$. In particular, $a(\bar{y}_v) = 0$ in iteration $\bar{x}_{i-1}$. But, in iteration $\bar{x}_{i-1}$, Case 2 holds. By line 9, $0 < a(\bar{x}_{i-1})$, but, by line 10, $0 = a(\bar{y}_v) \ge a(\bar{x}_{i-1})$, which is a contradiction.

Next, consider the case of $i < r$. If $\bar{x}_{i-1} \le x_{i+1}$, then x_{i+1} is adjacent to $\bar{x}_i$ by (3.2). Consider the increasing path $z \equiv x_r, x_{r-1}, \ldots, x_{i+1}, \bar{x}_i$. By Lemma 4.10, $b(\bar{x}_i) \le r - i \le k - i = a^*(\bar{x}_i)$ in iteration $\bar{x}_i$. When Case 1 holds in iteration $\bar{x}_i$, $a^*(\bar{x}_i) = 0$ and so $k = r = i$, which contradicts $i < r$. When Case 2 holds in iteration $\bar{x}_i$, $a(\bar{x}_i) = b(\bar{x}_i) = k$ and so $i = 0$, which contradicts $i \ge 1$. If $\bar{x}_{i-1} > x_{i+1}$, then $b(x_i) \le k - i$ after iteration x_{i+1} by Lemma 4.10. In particular, $b(x_i) \le k - i$ in iteration $\bar{x}_{i-1}$. But Case 2 of the algorithm holds in iteration $\bar{x}_{i-1}$. By line 10, $k - (i-1) = a(\bar{x}_{i-1}) \le b(x_i) \le k - i$, which is a contradiction.

So, in any case $z \ne \bar{y}$, i.e., $z < \bar{y}$. Similarly, $z < \bar{x}$. Consider the two increasing paths P_1 and P_2, where P_1 is $z \equiv x_r, x_{r-1}, \ldots, x_i, \bar{x}_{i^*}, \bar{x}_{i^*+1}, \ldots, \bar{x}_u \equiv \bar{x}$ (with $\bar{x}_{i^*}$ omitted when $x_i = \bar{x}_{i^*}$ and $i = i^*$) and P_2 is $z \equiv y_t, y_{t-1}, \ldots, y_j, \bar{y}_{j^*}, \bar{y}_{j^*+1}, \ldots, \bar{y}_v \equiv \bar{y}$ (with $\bar{y}_{j^*}$ omitted when $y_j = \bar{y}_{j^*}$ and $j = j^*$). By (3.1), $(\alpha, \beta) \in E$, where α is the second vertex of P_1 and β the second vertex of P_2. Note that $\alpha = x_p$ with $p \le r-1$ or $\alpha = \bar{x}_p$ with $p \le u$, and $\beta = y_q$ with $q \le t-1$ or $\beta = \bar{y}_q$ with $q \le v$. By Lemma 4.8 and the sentence just before Theorem 4.7, α $(k-1)$-neighborhood-covers (x, x') or β $(k-1)$-neighborhood-covers (y, y'), and so α or β k-neighborhood-covers both (x, x') and (y, y), contradicting the choice of z, except when $u = v = k$, $\alpha = \bar{x}_u$, $\beta = \bar{y}_v$, and $|P_1| = |P_2| = 1$. For the exceptional case, since either $i = i^*$ with $x_i = \bar{x}_i$ or $i = i^* + 1$ with $\bar{x}_{i-1} < x_i < \bar{x}_i$ form a clique, we have $r = i = i^* = u - 1 = k - 1$ or $r = i = i^* + 1 = u = k$ and so $\bar{x}_{k-1} \le z < \bar{x}_k$, i.e., $z, \bar{x}_k \in N^+[\bar{x}_{k-1}]$. Similarly, $z, \bar{y}_k \in N^+[\bar{y}_{k-1}]$. Without loss of generality, assume $\bar{x}_{k-1} \le \bar{y}_{k-1}$. Then, by (3.2), $\bar{y}_{k-1}$ is adjacent to $\bar{x}_k$. Now, $\bar{x}_k$ k-neighborhood-covers (x, x'). Also, $\bar{y}_{k-1}$ $(k-1)$-neighborhood-covers (y, y') and so $\bar{x}_k$ k-neighborhood-covers (y, y'). So we have $\bar{x}_k > z$ and $\bar{x}_k$ k-neighborhood-covers both (x, x') and (y, y'), which is a contradiction to the choice of z. ∎

THEOREM 4.12. *Algorithm CI finds a minimum k-neighborhood-covering set C^* and a maximum k-neighborhood-independent set I^* of a strongly chordal graph $G = (V, E)$ in linear-time if a strong elimination order is given.*

Proof. The correctness of the algorithm follows from Theorems 4.4, 4.7, and 4.11. The algorithm is linear, since iteration i costs only $O(|N^+(i)|)$ time excepting line 20, and line 20 in the whole algorithm costs at most $O(|E|)$ time. ∎

Acknowledgments. The authors thank referees for many constructive suggestions on the revision of this paper. In particular, one referee suggested the linear programming viewpoint as added at the end of section 1. A significant simplification of the proofs in section 4 was made after their comments.

REFERENCES

[1] R. P. Anstee, *Properties of* (0,1)*-matrices without certain configurations*, J. Combin. Theory Ser. A, 31 (1981), pp. 256–269.

[2] R. P. Anstee and M. Farber, *Characterization of totally balanced matrices*, J. Algorithms, 5 (1984), pp. 215–230.

[3] G. J. Chang, *Labeling algorithms for the domination problems in sun-free chordal graphs*, Discrete Appl. Math., 22 (1988/89), pp. 21–34.

[4] G. J. Chang and G. L. Nemhauser, *The k-domination and k-stability problems on sun-free chordal graphs*, SIAM J. Algebraic Discrete Methods, 5 (1984), pp. 332–345.

[5] G. J. Chang and G. L. Nemhauser, *Covering, packing and generalized perfection*, SIAM J. Algebraic Discrete Methods, 6 (1985), pp. 109–132.

[6] G. J. Chang, M. Farber, and Z. Tuza, *Algorithmic aspects of neighborhood numbers*, SIAM J. Discrete Math., 6 (1993), pp. 24–29.

[7] M. Farber, *Characterizations of strongly chordal graphs*, Discrete Math., 43 (1983), pp. 173–189.

[8] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[9] A. J. Hoffman, A. W. J. Kolen, and M. Sakarovitch, *Totally balanced and greedy matrices*, SIAM J. Algebraic Discrete Methods, 6 (1985), pp. 721–730.

[10] R. Laskar and D. Shier, *Construction of (r, d)-invariant chordal graphs*, Congressus Numerantium, 33 (1981), pp. 155–165.

[11] J. Lehel and Z. Tuza, *Neighborhood perfect graphs*, Discrete Math., 61 (1986), pp. 93–101.

[12] A. Lubiw, *Doubly lexical ordering of matrices*, SIAM J. Comput., 16 (1987), pp. 854–879.

[13] R. Paige and R. E. Tarjan, *Three partition refinement algorithms*, SIAM J. Comput., 16 (1987), pp. 973–989.

[14] E. Sampathkumar and P. S. Neeralagi, *The neighborhood number of a graph*, Indian J. Pure Appl. Math., 16 (1985), pp. 126–132.

[15] P. J. Slater, *R-domination in graphs*, J. Assoc. Comput. Mach., 23 (1976), pp. 446–450.

[16] J. P. Spinrad, *Doubly lexical ordering of dense* 0-1 *matrices*, Inform. Process. Lett., 45 (1993), pp. 229–235.

[17] J. Wu, *Neighborhood-Covering and Neighborhood-Independence in Strongly Chordal Graphs*, manuscript.