Graph Searching on Chordal Graphs

Sheng-Lung Peng*
National Tsing Hua University, Taiwan
e-mail: dr818303@cs.nthu.edu.tw

Ming-Tat Ko
Academia Sinica, Taiwan
e-mail: mtko@iis.sinica.edu.tw

Chin-Wen Ho
National Central University, Taiwan
e-mail: hocw@csie.ncu.edu.tw

Tsan-sheng Hsu*
Academia Sinica, Taiwan
e-mail: tshsu@iis.sinica.edu.tw

Chuan-Yi Tang
National Tsing Hua University, Taiwan
e-mail: cytang@cs.nthu.edu.tw

January 31, 1997

Abstract

In the graph searching problem, initially a graph with all edges contaminated is presented. We would like to obtain a state of the graph in which all edges are simultaneously clear by a sequence of moves using searchers. The objective is to achieve the desired state by using the least number of searchers. Two variations of the graph searching problem are considered in this paper. One is the edge searching, in which the clearing of an edge is accomplished by moving a searcher along the edge, and the other is the node searching, in which an edge is cleared by concurrently having searchers on both of its endpoints.

In this paper, we present a uniform approach to solve the above two graph searching problems on several classes of chordal graphs. For edge searching problem, we give an $O(mn^2)$-time algorithm on split graphs, an $O(m+n)$-time algorithm on interval graphs, and an $O(mn^k)$-time algorithm on k-starlike graphs (a generalization of split graphs), for a fixed $k \geq 2$, where m and n are the numbers of edges and vertices in the input graph, respectively. There are no polynomial algorithms known previously for any of the above problems. As a by-product, we also show that the edge searching problem remains NP-complete on chordal graphs. For node searching problem, we give an $O(mn^k)$-time algorithm on k-starlike graphs for a fixed k. This result implies that the pathwidth problem on k-starlike graphs is also in this time bound which greatly improves the previous results.

*Part of this research was supported by NSC85-2213-E-001-003.

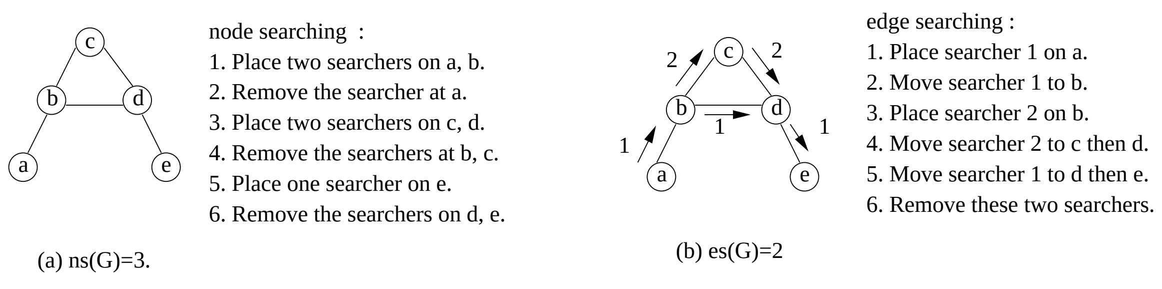
1 Introduction

The graph searching problem was first proposed by Parsons [Pa76]. A graph represents a system of tunnels. Initially, all edges of the graph are contaminated by a gas. We would like to obtain a state of the graph in which all edges are simultaneously clear by a sequence of moves using searchers. The objective is to achieve the desired state by using the least number of searchers.

In this paper, two versions of graph searching problem, the edge searching problem [MHGJP88] and the node searching problem [KP86], are discussed. These two problems are different in the ways of how searchers are moved in the graph and how contaminated edges are cleared. In *node searching*, the allowable moves are (1) placing a searcher on a vertex and (2) removing a searcher from a vertex. In node searching, a contaminated edge is *cleared* if both its two endpoints contain searchers. In *edge searching*, besides the allowable moves in the node-searching, one more move, (3) moving a searcher along an edge, is allowed. In edge searching, there are two ways in which a contaminated edge becomes clear: (i) if one endpoint of a contaminated edge contains a searcher, it can be cleared by moving a second searcher from that endpoint to the other endpoint along the edge, (ii) if one endpoint of a contaminated edge contains a searcher and all the other edges incident to it are cleared, the searcher can be moved along the edge to the other endpoint to clear the edge.

A vertex is *guarded* if it contains a searcher. A path that contains a guarded vertex is also called *guarded*. A cleared edge may be *recontaminated* once there is an unguarded path connecting the edge with a contaminated one.

An *edge-search strategy* (respectively, *node-search strategy*) of a graph G is a sequence of allowable moves in edge searching (respectively, node searching) that clears the initially contaminated graph. A search strategy is *optimal* if the maximum number of searchers on G at any time step is minimum over all search strategies of G. In edge searching (respectively, node searching), this number is called the *edge-search number* (respectively, *node-search number*) of G, and is denoted by $es(G)$ (respectively, $ns(G)$). For example, see the Figure 1.



node searching :
1. Place two searchers on a, b.
2. Remove the searcher at a.
3. Place two searchers on c, d.
4. Remove the searchers at b, c.
5. Place one searcher on e.
6. Remove the searchers on d, e.

(a) ns(G)=3.

edge searching :
1. Place searcher 1 on a.
2. Move searcher 1 to b.
3. Place searcher 2 on b.
4. Move searcher 2 to c then d.
5. Move searcher 1 to d then e.
6. Remove these two searchers.

(b) es(G)=2

Figure 1: Node-search and edge-search strategies.

For edge searching, it was shown in [La93] and [BS91] that there always exists an optimal edge-search strategy for G with $es(G)$ searchers that does not involve recontamination of any edges. It was shown that the edge searching problem is NP-complete on general graphs and can be solved in linear time on trees [MHGJP88]. This problem remains NP-complete for planar graphs with maximum vertex degree three [MS88].

For node searching, it was also shown in [KP86] and [BS91] that recontamination does not help in node-searching a graph with $ns(G)$ searchers. Note that the gate matrix layout problem [Mo90], the interval thickness problem [KP85], the pathwidth problem [RS83], the narrowness problem [KT92], and the vertex separation problem [Ki92] are all equivalent to the node searching problem. These problems have been extensively studied on many special classes of graphs. The node searching problem is NP-complete on planar graphs with vertex degree at most three [KP86], starlike graphs (a proper subclass of the chordal graphs) [Gu93], bipartite graphs [Kl93], cobipartite graphs (complement of bipartite graphs) [ACP87], and bipartite distance hereditary graphs (a proper subclass of the chordal bipartite graphs) [KBMK93]. There are linear-time algorithms on trees [Sc90][Mo90][EST94], cographs [BM93], and k-tree [BK96] for fix k, an $O(pn)$-time algorithm on permutation graphs [BKK93] (where p represents the pathwidth, i.e. node-search number minus one, of the input graph), and an $O(n^{2k+1})$-time algorithm on k-starlike graphs [Gu93] for fix k, where n is the number of vertices in the input graph.

In this paper, we use a uniform high-level approach to solve the graph searching problem on several special classes of chordal graphs. In particular, for the edge searching problem, we give an $O(mn^2)$-time algorithm on split graphs, an $O(m+n)$-time algorithm on interval graphs, and an $O(mn^k)$-time algorithm on k-starlike graphs for any fixed $k \geq 2$, where m and n are the numbers of edges and vertices in the input graph, respectively. All of the above problems were not previously known to be in polynomial time. In addition, we also provide an $O(mn^k)$-time and $O(m)$-space algorithm for the node searching problem on k-starlike graphs for any fixed k, which greatly improves a previous $O(n^{2k+1})$-time and $O(n^{2k+1})$-space algorithm in [Gu93]. We also show that the edge searching problem is NP-complete on chordal graphs.

The remains of this paper are organized as follows: In Section 2 we give some definitions and fundamental properties for this paper. As a by product, we also propose a linear time algorithm for recognizing k-starlike graphs in this section. The results of edge searching problem on split graphs, interval graphs, and k-starlike graphs are presented in Sections 3, 4, and 6 respectively. The results of node searching problem on k-starlike graphs are presented in Section 5. Finally, we give the conclusion in Section 7.

2 Preliminaries

2.1 Definitions

Let $G = (V, E)$ be a finite, simple and undirected graph, where V and E denote the vertex set and the edge set of the graph G respectively. Let $n = |V|$ and $m = |E|$. Let $N(v) =$

$\{w \mid (v,w) \in E\}$ denote the neighborhood of the vertex v and $\deg(v) = |N(v)|$. The *close neighborhood* of v is denoted as $N[v] = N(v) \cup \{v\}$. For vertex sets X and Y, let $N(X) = \{w \mid (v,w) \in E,\ v \in X, w \notin X\}$ and let $N_Y(X) = N(X) \cap Y$. For convenience, we use $N(v,w)$ to denote $N(\{v,w\})$. Let $G(X)$ be the subgraph of G induced by X. Throughout this paper, unless specified otherwise, an upper case letter denotes a set and a lower case letter denotes a singleton, for example, a vertex. For any vertex set with a given degree constraint (e.g., $< c$ and $> c$ where c is a constant), we will denote it in the right upper side of its set notation. For example, $I^{<c} = \{w \mid w \in I \text{ and } \deg(w) < c\}$ and $N_Y^{>c}(X) = \{w \mid w \in N_Y(X) \text{ and } \deg(w) > c\}$.

A *clique* in a graph G is a complete subgraph of G. A clique, also refers to a set of vertices, whose induced subgraph is complete when there is no confusion in the description. An *independent* set in a graph G is a vertex subset I in G such that no two vertices in I are adjacent in G.

A graph G is called a *split graph* if its vertex set V can be partitioned into two subsets C and I such that C induces a maximal clique and I is an independent set of G. Throughout this paper, we use $G = (C \cup I, E)$ to denote a split graph in which C induces a maximal clique and I is an independent set. Split graphs can be recognized in linear time [HS81].

A graph G is a *chordal* graph if there is a tree T and a family of subtrees of T indexed by the vertices of G, $\{F_u\}_{u \in V}$, such that subtrees F_u and F_v share a node of T if and only if the vertices u and v are adjacent. Furthermore, it is possible to construct T in such a way that there is a bijection between the nodes of T and the maximal cliques of G, where the subtree F_u is comprised of all nodes that correspond to maximal cliques containing the vertex u [Ga74]. In general, such a tree T is called a *clique tree model* or *representation* for G.

A graph is an *interval graph* if the set of maximal cliques can be linearly ordered such that for any vertex v, the cliques containing v occur consecutively in the linear ordering [Gi64]. Thus a chordal graph is an interval graph if and only if one of its clique trees is a path.

A chordal graph G is called *starlike* if and only if one of its clique trees is a star. Let $\{X_0, X_1, \ldots, X_r\}$ be the set of all maximal cliques of a starlike graph. Throughout this paper, X_0 denotes the clique corresponding to the central node of the star clique tree of G. X_0 is called the *central clique* and the other maximal cliques X_i's$(i \neq 0)$ are called *peripheral cliques*. A peripheral clique X of a starlike graph is called an i-p-clique if $|X \setminus X_0| = i$. We call the vertices in X_0 (respectively, $V \setminus X_0$) the *central* (respectively, *peripheral*) *vertices* of G. For an integer k, we call a starlike graph G *k-starlike* if $\max_{i=1}^{r} |X_i \setminus X_0| = k$. Note that this implies that split graphs are 1-starlike graphs.

For any k-starlike graph G with maximal cliques $\{X_0, X_1, \ldots, X_r\}$, the peripheral vertices in X_i have the same close neighborhood for each i, $1 \leq i \leq r$. Thus we can recognize k-starlike graphs as follows: Firstly, we classify the peripheral vertices in G according to their neighbors. All peripheral vertices in the same class have the same set of neighbors. This can be done in linear time by a slightly modified lexicographic breadth-first search algorithm of [Go80]. All peripheral vertices in the same class can be contracted into one vertex. After contraction, the resulting graph is a split graph. By checking the maximum cardinality among classes of vertices corresponding to peripheral vertices in the contracted graph, we can decide the value k for a k-starlike graph. Thus k-starlike graphs can also be recognized in linear time.

2.2 Properties

During an edge-searching of any graph, for any time step, there are three types of vertices: clear, contaminated, and partially clear [La93]. A vertex is called *clear* if all of its incident edges are cleared, *contaminated* if all of its incident edges are contaminated, and *partially clear* if there are both contaminated and clear edges incident to it. A partially clear vertex is called a *spot* vertex if it has only one contaminated incident edge and is called a *stain* vertex otherwise. In [La93], LaPaugh defined the *clearing move* composed of several allowable moves. A clearing move clears an edge and reaches a state that satisfied the following two conditions: (I) no clear or contaminated vertex contains a searcher; and (II) no vertex contains more than one searcher.

The clearing moves are of the following two types for an edge (x, y):

1. $m^+(x, y)$: Vertex x has at least two incident contaminated edges. If vertex x contains no searcher, we first place a searcher on x. Then move a second searcher from x to y along the edge (x, y). Remove searchers from y if vertex y violates conditions (I) or (II).

2. $m^-(x, y)$: Vertex x has only one incident contaminated edges. If vertex x contains no searcher, we first place a searcher on x. Then move the searcher at x to y along the edge (x, y). Remove searchers from y if vertex y violates conditions (I) or (II).

Note that a clearing move clears exactly one edge and it does not cause any recontamination. In other words, any edge-search strategy composed of clearing moves has $|E|$ clearing moves. LaPaugh proved the following important theorem that implies recontamination can not help in the optimal edge-search strategy.

Theorem 2.1 *[La93]There is an optimal edge-search strategy composed of clearing moves.*

By Theorem 2.1, in the following we only consider edge-search strategies composed of clearing moves.

In an edge-search strategy composed of clearing moves, the state at time step t is defined as $S_t = (Y_t, Z_t)$ where $Y_t \subseteq V$ is the set of guarded vertices and $Z_t \subseteq E$ is the set of remaining contaminated edges. An edge-search strategy $\mathcal{S}$ of G can be represented as a sequence of states S_i's and clearing moves m_i's:

$$S_0, m_1, S_1, ..., m_{|E|}, S_{|E|},$$

where $S_0 = (Y_0 = \emptyset, Z_0 = E)$, $S_{|E|} = (Y_{|E|} = \emptyset, Z_{|E|} = \emptyset)$, and state S_i is obtained from S_{i-1} by applying clearing move m_i for $i = 1, ..., |E|$. A vertex u is *clear* in a state S_t if all its incident edges are not in Z_t. We say a vertex u is *cleared* at S_t if it is the first state u becomes clear.

With respect to a state, any searcher not guarding a vertex is called a *free* searcher. If the number of searchers to clear a graph is given, a *deadlock state* in a search strategy is a state with no free searcher and every guarded vertex is a stain vertex. Note that there is no deadlock state in an optimal edge-search strategy given $es(G)$ searchers.

The contaminated edge (x, y) incident to a spot vertex x can be cleared by $m^{-}(x, y)$ in which no free searcher is used. The following lemma can be obtained straightforward.

Lemma 2.2 *There exists an optimal edge-search strategy $S_0, m_1, S_1, ..., m_{|E|}, S_{|E|}$ such that for every S_i containing spot vertices, $m_{i+1} = m^{-}(x, y)$, where x is a spot vertex and (x, y) is the contaminated edge incident to x.*

Notice that, if there are many spot vertices in a state, their incident edges can be cleared in an arbitrary order. In the rest of this paper, all the edge-search strategies considered satisfy Lemma 2.2.

In the remaining of this section, we state some basic lemmas about the graph searching problem which are useful in understanding our strategy.

Lemma 2.3 *[Pa76] For any clique $K_n, n \geq 4$, $es(K_n) = n$.*

Lemma 2.4 *Let K_n be a clique, $n \geq 4$. If $W \subset V(K_n)$, $|W| \geq 2$, and all the vertices in $V(K_n) \setminus W$ are guarded, then K_n can be cleared by additional $|W|$ searchers without moving any guarding searchers in $V(K_n) \setminus W$.*

Proof. The proof is given by the following edge-search strategy with additional $|W|$ searchers which does not move searchers guarding vertices in $V(K_n) \setminus W$.

1. Place $|W| - 1$ searchers on W except one vertex, say v. Then clear all contaminated edges in K_n except those incident to v by the remaining free searcher.

2. Since $|W| \geq 2$, there exists a vertex $u \neq v$ in W. First apply a clearing move $m^{-}(u, v)$ to clear the edge (u, v). Then clear all contaminated edges incident to v by the remaining free searcher. Now that all vertices in K_n are cleared. **Q.E.D.**

The following lemmas play an important role in our result.

Lemma 2.5 *Let $G = (V, E)$ be a starlike graph with maximal cliques $\{X_0, X_1, \ldots, X_r\}$ and $es(G) = |X_0| \geq 4$. Then there exists an optimal edge-search strategy such that the last cleared edge incident to the first cleared vertex in X_0 is in the central clique.*

Proof. Let us consider an arbitrary optimal edge-search strategy $\mathcal{S}$. In $\mathcal{S}$, let u be the first cleared vertex in X_0 and let (u, a) be the last cleared edge incident to u. If $a \in X_0$, $\mathcal{S}$ is the desired optimal edge-search strategy. In the following, assume $a \notin X_0$. Note that a has to be a 1-p-clique. Let w be the last guarded vertex in X_0 and S_t be the first state w is guarded. Since u is the first cleared vertex in X_0 and only $|X_0|$ searchers are used in $\mathcal{S}$, u must be cleared by the following steps: S_{t-1}, $m^+(u, w)$, S_t, $m^-(u, a)$, S_{t+1}. We have the following three cases:

1. $\deg(a) = 1$. We modify $\mathcal{S}$ into $\mathcal{S}'$ by changing the clearing moves from S_{t-1} to S_{t+1} by S_{t-1}, $m^+(u, a)$, S'_t, $m^-(u, w)$, S_{t+1}. The new strategy $\mathcal{S}'$ uses the same number of searchers as $\mathcal{S}$ and it is also optimal.

2. $\deg(a) = 2$. Let v be the other vertex in X_0 adjacent to a. By Lemma 2.2, a must be cleared after S_{t+1}, $m^-(a, v)$, S_{t+2}. We modify $\mathcal{S}$ into $\mathcal{S}'$ by changing the clearing moves from S_{t-1} to S_{t+2} by S_{t-1}, $m^+(u, a)$, S'_t, $m^-(a, v)$, S'_{t+1}, $m^-(u, w)$, S_{t+2}. It is easy to see $\mathcal{S}'$ is a desired optimal edge-search strategy.

3. $\deg(a) \geq 3$. Since u is the first cleared vertex in X_0, there is no spot vertex in G at S_{t-1} by Lemma 2.2. So S_{t+1} is a deadlock state which is impossible. **Q.E.D.**

The relations between the edge-search number and the node-search number of a graph are described in the following lemma.

Lemma 2.6 *[KP86] For any graph G, $ns(G) - 1 \leq es(G) \leq ns(G) + 1$.*

3 Edge searching on split graphs

In this section, we discuss the edge searching problem on split graphs. The following lemma is implicitly derived from [Gu93].

Lemma 3.1 *[Gu93] If $G = (C \cup I, E)$ is a split graph, then $ns(G) = |C|$ or $|C| + 1$.*

According to Lemmas 2.3, 2.6 and 3.1, we can easily obtain the following lemma.

Lemma 3.2 *If $G = (C \cup I, E)$ is a split graph with $|C| \geq 4$, then $es(G) = |C|$, $|C| + 1$ or $|C| + 2$.*

In the case of $|C| = 2$, if G is a path then $es(G) = 1$; otherwise, $es(G) = 2$.

Lemma 3.3 *If $G = (C \cup I, E)$ is a split graph with $|C| = 3$, then the followings are true.*

(1) *$es(G) = 2$ if and only if $\exists w \in C$, $\deg(w) = 2$ and $\forall u, v \in C$, $|N(u) \cap N(v)| \leq 1$.*

(2) *$es(G) = 4$ if and only if $\forall u, v \in C$, $|N_I(u) \cap N_I(v)| \geq 2$.*

(3) *$es(G) = 3$, if none of the conditions specified in (1) and (2) is satisfied.*

Proof. (1) Since $|C| = 3$ (not a path), so $es(G) > 1$. We first consider the "only if" part. The graphs depicted in Fig. 2(a) and 2(b) are the minimal graphs not satisfying the specified condition. It can be verified that their edge-search numbers are greater than two. Thus, $es(G) = 2$ implies the graph G satisfies the specified condition.

We now prove the "if" part. A graph G satisfying the specified condition is shown in Fig. 2(c). All the vertices not in C is of degree one and adjacent to vertices u or v. We can clear G by the following strategy: We first place one searcher on u and clear all vertices in $N_I(u)$ by the other free searcher. Then we apply $m^+(u, w)$ and $m^-(w, v)$. Finally, we apply $m^-(u, v)$ and clear all vertices in $N_I(v)$ by this searcher. Hence $es(G) = 2$.

(2) We first prove the "only if" part by contradiction. Let $C = \{u, v, w\}$ and $N_I(u) \cap N_I(v) = \{x\}$. The following strategy clears G by using three searchers. Firstly, we place two searchers on u and w respectively. Next we clear the edge (u, w) and all vertices in $I \backslash N_I(v)$ by using the third searcher. Then there are two uncleared paths from u to v. We first apply $m^+(u, v)$. Then $m^-(u, x)$ and $m^-(x, v)$. Finally, we can clear all uncleared edges by one free searcher. Thus three searchers are sufficient.

For the "if" part, it is obvious that the graph G can be cleared by using four searchers. The graph depicted in Fig. 2(d) is the minimal graph satisfying the specified condition. It can be verified that at least four searchers are needed to clear it. Thus, four searchers are necessary and $es(G) = 4$.

(3) The correctness follows from (1) and (2). **Q.E.D.**

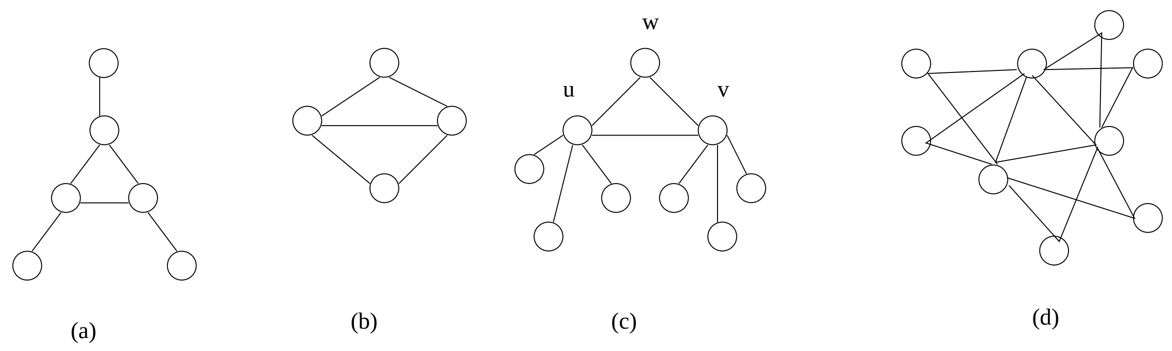


Figure 2: The sample graphs for Lemma 3.3.

In the following, we only consider the split graph with the maximum clique size greater than three. The following two lemmas state the necessary and sufficient conditions for G to be of $es(G) = |C|$ and $es(G) = |C| + 1$, respectively.

Lemma 3.4 *Let $G = (C \cup I, E)$ be a split graph with $|C| \geq 4$. Then $es(G)$=$|C|$ if and only if $\exists u, v \in C$, such that the following conditions hold.*

(1) $|C \backslash N(N_I^{\geq 2}(u, v) \backslash (A \cup W))| \geq 2$, and

(2) $|C\backslash N(N_I(u) \cup N_I^{>2}(v)\backslash(B \cup W))| \geq 1$*,*

where (i) $A \subseteq N_I^{>2}(u,v)$ *and* $|A| \leq 1$*, (ii)* $W \subseteq N_I^{>2}(v)\backslash N_I^{>2}(u)$ *and* $|W| \leq 1$*, and (iii)* $B \subseteq N_I^{=2}(u)$ *and* $|B| \leq 1$*.*

Proof. Suppose that $es(G) = |C|$. Consider an optimal edge-search strategy, $\mathcal{S} = S_0$, m_1, S_1, …, $m_{|E|}$, $S_{|E|}$ such that $\mathcal{S}$ satisfies Lemma 2.5 and in which u and v are the first and the second cleared vertex in C respectively, and y and x are the last and the second last guarded vertex in C respectively. Let S_{t_1} (respectively, S_{t_2}) be the first state in which x (respectively, y) is guarded. Let S_{t_3} (respectively, S_{t_4}) be the first state in which u (respectively, v) is cleared.

At S_{t_1}, since there are $|C|-1$ searchers on the central clique, at most one peripheral vertex is guarded. If such a peripheral vertex z exists, z has to be a spot vertex and by Lemma 2.2, z is cleared at S_{t_1+1}. Otherwise, z is a stain vertex and then there is no free searcher between S_{t_1} and S_{t_3}. In order that u can be cleared, edge (u,x) has to be cleared just before S_{t_1} and the scenario must be $m^+(u,x), S_{t_1}, m^-(u,y), S_{t_2}(= S_{t_3})$. Then every vertex w in $C \setminus \{u\}$ is a stain vertex, since at least (w,x) and (w,y) are contaminated. Thus, S_{t_2} is a deadlock state which leads to a contradiction.

Between S_{t_1} and S_{t_4}, since at most one free searcher can be used, no peripheral vertices at S_{t_1} of degree greater than two except z are cleared. Hence only peripheral vertices of degree no greater than two are cleared between S_{t_1+1} and S_{t_2}. Since $\mathcal{S}$ satisfies Lemma 2.5, S_{t_3} is obtained from S_{t_2} by applying $m^-(u,y)$ or $t_2 = t_3$. In either case, we obtain that all the vertices in $N_I(u)$ are already clear in S_{t_3-1}. Otherwise, there is a guarded vertex $z' \in N_I(u)$ with (z',u) being clear at S_{t_3}. If $deg(z') = 2$, by Lemma 2.2, we may change $\mathcal{S}$ to another optimal edge-search strategy such that z' is cleared earlier than u. If $deg(z') \geq 3$, then (z',u) is cleared just before S_{t_3-1}. In S_{t_3-2}, u and z' both are stain vertices. To continue the strategy, v must be a spot vertex in S_{t_3}, which implies v has been a spot vertex at S_{t_3-2}. It contradicts to that the edge-search strategy satisfying Lemma 2.2.

Let $W = \{w | w \in N_I^{>2}(v)\backslash N_I^{>2}(u)$ and w is contaminated in $S_{t_1}\}$, $A = \{a | a \in N_I^{>2}(u,v)\backslash W$ and a is not clear in $S_{t_1-1}\}$ and $B = \{b | b \in N_I^{=2}(u)$ and b is not clear in $S_{t_2-1}\}$. By the definition of W, A and B and the fact that vertices in $N_I^{>2}(u)$ are clear in S_{t_1+1}, vertices in $N_I^{>2}(v) \setminus W$ are clear in S_{t_1+1}, vertices in $N_I^{>2}(u,v) \setminus (W \cup A)$ are clear in S_{t_1-1} and vertices in $N_I(u) \setminus B$ are clear in S_{t_2-1}. Thus, vertices in $N_I(u) \cup N_I^{>2}(v) \setminus (B \cup W)$ are clear in S_{t_2-1}. Therefore, x and y is not in $N(N_I^{>2}(u,v)\backslash(W \cup A))$ and y is not in $N(N_I(u) \cup N_I^{>2}(v)\backslash(B \cup W))$. Namely, condition (1) and (2) hold. To complete the proof, it remains to prove that $|W| \leq 1$, $|B| \leq 1$ and $|A| \leq 1$.

Since v is cleared at S_{t_4}, the edges between v and vertices in W are not in Z_{t_4}. Thus, those edges are only cleared between S_{t_3} and S_{t_4} by at most two searchers, a free searcher (from u) and the searcher on v, that implies $|W| \leq 2$. Suppose $W = \{w_1, w_2\}$. Let $e = (v, w_2)$ be the last cleared edge incident to v. Then the clearing scenario before S_{t_4} has to be S_{t_4-2}, $m^+(v,w_1)$, S_{t_4-1}, $m^-(v,w_2)$, S_{t_4}. In S_{t_4}, w_1 and w_2 are both stain vertices. Since $\mathcal{S}$ satisfies

Lemma 2.2, every vertex in $C \setminus \{u, v\}$ is a stain vertex. Thus S_{t_4} is a deadlock state which is a contradiction. Hence we conclude that $|W| \leq 1$.

Since $B \subset N_I(u)$, vertices in B must be cleared in S_{t_2}. From S_{t_2-1} to S_{t_2}, there is at most one vertex becomes clear. Thus, $|B| \leq 1$.

By definition of A, vertices in A are not clear at S_{t_1-1}, but are clear at S_{t_1+1}. Since there is at most one peripheral vertex of degree greater than two becomes clear from S_{t_1-1} to S_{t_1+1}, we conclude that $|A| \leq 1$.

Conversely, assume that conditions (1) and (2) hold. Let $C_1 = C \setminus N(N_I^{>2}(u, v) \setminus (A \cup W))$ and $C_2 = C \setminus N(N_I(u) \cup N_I^{>2}(v) \setminus (B \cup W))$. Note that $C_2 \subseteq C_1$ and by the conditions we have two central vertices x and y such that $x, y \in C_1$ and $y \in C_2$. Without loss of generality, assume that $A = \{a\}$, $W = \{w\}$, $B = \{b\}$ and $(a, x), (b, y) \in E$. We provide the following edge-search strategy using exactly $|C|$ searchers to complete the proof.

1. Place $|C| - 2$ searchers on $C \setminus \{x, y\}$ as guards and clear all peripheral vertices in $I \setminus N_I(x, y)$. Now, all vertices in $N_I^{>2}(u, v) \setminus \{a, w\}$ are clear.

2. Clear vertex a: From condition (2), $(a, y) \notin E$. At this step, we have two free searchers. Apply $m^+(u, a)$ or $m^+(v, a)$ first. Then use the second free searcher to clear all edges incident to a with (a, x) be the last cleared edge. Now, a is clear and x is guarded.

3. Clear edges in $G(C \setminus \{y\})$ and uncleared vertices in $N_I^{<3}(u) \setminus B$, by the remaining free searcher.

4. Clear vertex b: Apply $m^+(u, b)$ first then $m^-(b, y)$. At this state, all vertices in C are guarded and all edge incident to u except (u, y) are cleared.

5. Clear u: Apply $m^-(u, y)$. Now we have a free searcher again.

6. Clear all contaminated edges in $G(C)$ and all uncleared vertices in $I^{<3}$ by the free searcher.

7. Clear v and w: At this step, (v, w) is the only one uncleared edge incident to v. Apply $m^-(v, w)$ first and v is cleared. Then use the free searcher to clear all the remaining contaminated edges incident to w. Now, w is cleared and we have two free searchers.

8. Clear all the remaining uncleared peripheral vertices in $I^{>2}$ (in $N_I(x, y) \setminus N_I(u, v)$) by the two free searchers.

Thus, if conditions (1) and (2) hold then $es(G) = |C|$. **Q.E.D.**

Lemma 3.5 *Let $G = (C \cup I, E)$ be a split graph with $|C| \geq 4$ and $es(G) > |C|$. Then $es(G) = |C| + 1$ if and only if $\exists u, v \in C$ such that $|N_I^{>2}(u) \cap N_I^{>2}(v)| \leq 2$.*

Proof. Suppose that $es(G) = |C| + 1$. Let us consider an optimal edge-search strategy $\mathcal{S}$. Let u (respectively, v) be the first cleared (respectively, last guarded) vertex in C. Let S_t (respectively, $S_{t'}$) be the first state in which v (respectively, u) is guarded (respectively, cleared). Between S_{t+2} and $S_{t'-1}$, there is at most one free searcher and thus no peripheral vertices of degree greater than 2 are cleared. At S_{t+1} , at most one vertex in $N_I^{>2}(u) \cap N_I^{>2}(v)$ is cleared. At $S_{t'}$, at most two peripheral vertices are guarded, and all uncleared vertices in $N_I^{>2}(u) \cap N_I^{>2}(v)$ are guarded. It implies $|N_I^{>2}(u) \cap N_I^{>2}(v)| \leq 3$. In the case of $|N_I^{>2}(u) \cap N_I^{>2}(v)| = 3$, at $S_{t'}$, one vertex in $N_I^{>2}(u) \cap N_I^{>2}(v)$ has to be clear and each of the other two vertices is guarded with at least two contaminated incident edges. By Lemma 2.2 and the assumption of u, each vertex in $C \setminus \{u\}$ is a stain vertex at $S_{t'}$. Thus, $S_{t'}$ is a deadlock state that contradicts to $\mathcal{S}$ is an edge-search strategy. It follows that $|N_I^{>2}(u) \cap N_I^{>2}(v)| \leq 2$.

Now we assume that $|N_I^{>2}(u) \cap N_I^{>2}(v)| \leq 2$. Without loss of generality, let $N_I^{>2}(u) \cap N_I^{>2}(v) = \{a, b\}$. We provide the following edge-search strategy using $|C| + 1$ searchers to complete the proof.

1. Place $|C| - 1$ searchers on $C \backslash \{v\}$ and clear all vertices in $I \backslash N_I(v)$ by two free searchers.

2. Clear u and $\{a, b\}$: Firstly, we apply $m^+(u, a)$ and use the other free searcher to clear the edges incident to vertex a with (a, v) be the last cleared one. At this state, all vertices in C are guarded. We use the remaining free searcher to clear the all edges in the central clique. Then we apply $m^-(u, b)$ and clear all edges incident to b by the free searcher. After b is cleared, we have two free searchers again.

3. Clear the all vertices in $N_I(v) \backslash \{a, b\}$ by the two free searchers.

Thus G can be cleared using $|C| + 1$ searchers. **Q.E.D.**

By checking the conditions in Lemmas 3.4 and 3.5, we have an algorithm to determine $es(G)$ of a split graph G. The algorithm has two phases. First, for any pair of vertices u and v in C, we check the conditions in Lemma 3.4 to determine whether $es(G) = |C|$ or not. If the answer is no, then we go to the second phase and check the conditions in Lemma 3.5 to determine whether $es(G) = |C| + 1$ or not. If both checking fail, then by Lemma 3.2, $es(G) = |C| + 2$.

The time complexity depends on checking the conditions of Lemmas 3.4 and 3.5. The condition of Lemma 3.5 can be easily checked in $O(mn^2)$ time. Thus the bottleneck is to check the conditions of Lemma 3.4. A naive implementation of checking the conditions of Lemma 3.4 takes $O(mn^5)$ time. However, by the following observation and careful implementation, it also can be done in $O(mn^2)$ time.

For each pair $u, v \in C$, we want to find whether there exist sets A, B and W satisfy the conditions of Lemma 3.4 or not. If the sets A, B and W exist, then there exist two central vertices x and y, called an *indicator pair* for u and v, such that $x, y \in C \backslash N(N_I^{>2}(u, v) \backslash (A \cup W))$ and $y \in C \backslash N(N_I(u) \cup N_I^{>2}(v) \backslash (B \cup W))$. Instead of checking the existence of sets A, B and W directly, our algorithm is to check the existence of an indicator pair x, y for u, v. By a careful investigation, we can see that a central vertex c is a candidate of x if the following conditions hold:

(X1) $|N_I(c) \cap N_I^{\geq 2}(u)| \leq 1$,

(X2) $|N_I(c) \cap N_I^{\geq 2}(u, v)| \leq 2$,

and a central vertex c is a candidate of y if the following conditions hold:

(Y1) $|N_I(c) \cap N_I^{\geq 2}(u)| = 0$,

(Y2) $|N_I(c) \cap (N_I^{\geq 2}(v) \setminus N_I^{\geq 2}(u))| \leq 1$, and

(Y3) $|N_I(c) \cap N_I^{=2}(u)| \leq 1$.

Let c_x be a candidate of x and let c_y be a candidate of y. In the cases of $|N_I(c_x) \cap N_I^{\geq 2}(u, v)| \leq 1$ or $|N_I(c_y) \cap (N_I^{\geq 2}(v) \setminus N_I^{\geq 2}(u))| = 0$, c_x, c_y can serve as an indicator pair for u and v. In case of $|N_I(c_x) \cap N_I^{\geq 2}(u, v)| = 2$ and $|N_I(c_y) \cap (N_I^{\geq 2}(v) \setminus N_I^{\geq 2}(u))| = 1$, c_x, c_y can serve as an indicator pair for u and v only if $N_I(c_y) \cap (N_I^{\geq 2}(v) \setminus N_I^{\geq 2}(u)) \subset N_I(c_x) \cap N_I^{\geq 2}(u, v)$.

The outline of the algorithm to check the conditions of Lemma 3.4 is sketched in the following: Let $C = \{c_1, c_2, \ldots, c_{|C|}\}$. For each pair $u, v \in C$, we construct the sets $N_I^{\geq 2}(u, v)$, $N_I^{\geq 2}(v) \setminus N_I^{\geq 2}(u)$, and $N_I^{=2}(u)$. Let D and F be arrays such that $D[i]$ is the number of c_i's neighbors in $N_I^{\geq 2}(u, v)$ and $F[i]$ is the number of c_i's neighbors in $N_I^{=2}(u)$. We use D and F mainly to determinate the candidates of x and y. For those c_i's that can be a candidate of x or y, we use an auxiliary array D' to record the possible vertices of W and A. For example, if c_i can be a candidate of x, then $D'[2i] = (w_i, a_i)$ where w_i is a candidate in W and a_i is a candidate in A. If a_i can also be a candidate in W, the $D'[2i-1] = (a_i, w_i)$ is recorded. Note that W and A can be empty. In such a case that w_i and a_i will be replaced by zeros. It is not hard to see that if $D[i] > 2$, then c_i cannot be a candidate of x and y. Thus we only consider those c_i's with $D[i] \leq 2$.

For a vertex c_i with $D[i] = 2$, let w_i and a_i be the two vertices in $N_I(c_i) \cap N_I^{\geq 2}(u, v)$. By conditions (Y1) and (Y2), c_i can not be a candidate of y. Furthermore, if both w_i and a_i belong to $N_I^{\geq 2}(u)$, by condition (X1) c_i can not be a candidate of x. Otherwise, c_i is a candidate of x. In the case that both w_i and a_i are in $N_I^{\geq 2}(v) \setminus N_I^{\geq 2}(u)$, let $D'[2i] = (w_i, a_i)$ and $D'[2i-1] = (a_i, w_i)$. For the case that only one of w_i and a_i is in $N_I^{\geq 2}(v) \setminus N_I^{\geq 2}(u)$, say w_i, let $D'[2i] = (w_i, a_i)$.

For a vertex c_i with $D[i] = 1$, let $\{a_i\} = N_I(c_i) \cap N_I^{\geq 2}(u, v)$. By conditions (X1) and (X2), c_i is a candidate of x. In the case that $a_i \in N_I^{\geq 2}(u)$ (by condition (Y1)) and the case that $a_i \notin N_I^{\geq 2}(u)$ but $F[i] \geq 2$ (by condition (Y3)), c_i can not be a candidate of y. In these cases, let $D'[2i] = (0, a_i)$. In the case that $a_i \notin N_I^{\geq 2}(u)$ but $F[i] \leq 1$, then c_i can also be a candidate of y and let $D'[2i] = (a_i, 0)$ and $D'[2i-1] = (0, a_i)$.

Finally, for a vertex c_i with $D[i] = 0$, let $D'[2i] = (0, 0)$. If $F[i] \geq 2$, then by condition (Y3) c_i can not be a candidate of y but can be a candidate of x. If $F[i] \leq 1$ then c_i can be a candidate of y.

By the above discussion, we can classify the central vertices into three classes. One is the set C_x which denotes the set of candidate x but not y, another is C_y which denotes the set of candidate y, and the other is the set $C \backslash (C_x \cup C_y)$. Note that a candidate of y can also be a candidate of x. Hence the conditions of Lemma 3.4 hold if $|C_y| \geq 2$. If $|C_y| = 0$, then the conditions of Lemma 3.4 are unsatisfied. If $|C_y| = 1$, then let $c_i \in C_y$ and $D'[2i] = (w_i, 0)$ (or $(0, 0)$). Now we can check whether C_x exists a $c_j \neq c_i$ such that $D'[2j] = (0, 0)$, $(0, a_j)$, or (w_i, a_j). If it successes, then the conditions of Lemma 3.4 hold.

It is not hard to see that these steps can be done in $O(m + n)$ time. For $O(n^2)$ pairs of u and v, overall it takes $O(mn^2)$ time to check the conditions in Lemma 3.4. Thus we have the following theorem.

Theorem 3.6 *There is an $O(mn^2)$-time and $O(m)$-space algorithm to determine the edge-search number of a split graph.*

4 Edge searching on interval graphs

In this section, we will solve the problem of finding the edge-search number of an interval graph. A graph $G = (V, E)$ is called an *interval graph* if there is a family of intervals indexed by vertices in V, $F = \{I_v = [a_v, b_v] \mid v \in V\}$ such that $(u, v) \in E$ if and only if $I_u \cap I_v \neq \emptyset$. F is called an *interval model* of G. We say a_v and b_v are the *left endpoint* and the *right endpoint* of the interval I_v respectively. For simplicity, "interval" and "vertex" are coincident.

Another characteristic for G to be an interval graph is that the maximal cliques of G can be linearly ordered such that for every vertex v of G , the maximal cliques containing v occur consecutively [Gi64]. If these linearly ordered maximal cliques $C_1, C_2, \ldots, C_r$ of G are given (the ordering can be found in $O(m + n)$ time [BL76]), then according to this ordering, an interval model $F = \{[a_v, b_v] \mid v \in V\}$ can be obtained by letting a_v (respectively b_v) be the minimum (respectively maximum) index of the clique containing vertex v. In the following, we assume that such an interval model is given.

For any interval graph G, $ns(G)$ is trivially equal to its maximum clique size. By Lemma 2.3 and 2.6, we have the following lemma.

Lemma 4.1 *If $G = (V, E)$ is an interval graph with maximum clique size $\omega \geq 4$, then $es(G) = \omega$ or $\omega + 1$.*

We now give an algorithm for the case when $\omega \geq 4$. The case that $\omega < 4$ will be treated later. The algorithm scans intervals according to their right endpoints in non-decreasing order. If two intervals have the same right endpoints, then the interval with the smaller left endpoint is scanned first. (Intervals may have same left and right endpoints. In this case, our algorithm scans them in arbitrary order.)

Algorithm ESI

Input: A family of intervals F.
Output: $es(G)$

```
1.  Sort $F$ to $W$ according to their right endpoints, if there are some intervals with the same
    right endpoints, then they are sorted according to their left endpoints. /* The set $W$
    contains the all remaining uncleared intervals. */
2.  $var\_es = 0$ /* Initially there is no searcher on $G$. */
3.  DO WHILE $W \neq \emptyset$
4.      Let $u$ be the first interval in $W$.
5.      Let $U = \{v \mid v \in N(u) \cap W$ and $v$ is not guarded$\}$.
6.      Case 1: $U \neq \emptyset$. Let $v \in U$. /* Note that $N[u] \cap W$ is a clique. */
7.              $var\_es = \max(var\_es, |N[u] \cap W|)$
8.              Place searchers on $U \backslash \{v\}$ and a searcher on $u$ if $u$ is not guarded.
9.              Clear contaminated edges in $G(N[u] \cap W \backslash \{v\})$ by a free searcher.
10.             Move the searcher on $u$ to $v$.
11.             Clear contaminated edges in $G(N[u] \cap W)$ by a free searcher.
12.     Case 2: $U = \emptyset$ and $u$ has a searcher. /* $u$ is cleared. */
13.             Remove the searcher on $u$.
14.     Case 3: $U = \emptyset$ and $u$ has no searcher.
15.             $var\_es = \max(var\_es, |N[u] \cap W| + 1)$
16.             Place one searcher on $u$.
17.             Clear $u$ by using a free searcher.
18.     $W = W \backslash \{u\}$
19. ENDDO
20. $es(G) = var\_es$
```

END.

Since each interval u is scanned according to its right endpoint in non-decreasing order, such an ordering can be shown to be a perfect elimination ordering of G [Go80], which means that $N[u] \cap W$ forms a clique of G. Each time a vertex u is removed from W, it is cleared since its incident edges are all cleared and all its neighbors are cleared or guarded by searchers. This condition can be easily proved by induction on the ordering of u. The number of searchers used is recorded in the variable var_es. The only situation that the algorithm needs $\omega + 1$ searchers occurs when Case 3 holds and $K = N[u] \cap W$ is a maximum clique of G. Under this condition, we can find another two vertices u_1 and u_2 such that the algorithm scans u_1 and u_2 before and after u, respectively, $N[u_1] \cap K = N[u_2] \cap K = K - \{u\}$, and u_1, u_2, u are mutually nonadjacent. The existence of u_1 follows from the fact that $U = \emptyset$. If u_2 does not exist, then there exists a vertex v in K such that u and v have the same right endpoints, and the left endpoint of v is smaller than that of u. This implies v is scanned (and thus removed from W) before u by the algorithm, a contradiction. Now, the set $K \cup \{u_1, u_2\}$ induces a split graph that does not satisfy the conditions of Lemma 3.4. Hence, $\omega + 1$ searchers is necessary.

As for the time complexity of the algorithm, we can implement the algorithm in the following way: Sort all endpoints by a radix sort algorithm in $O(n)$ time (note that the values of endpoints are indices of maximal cliques of G and the number of maximal cliques is less than n). Then, process each endpoint in the sorted order. If it is the left endpoint of a vertex, mark the vertex as a candidate of the set $N[u] \cap W$. If it is the right endpoint of a vertex, then process the vertex as described in the algorithm. Hence, $O(n)$ time is sufficient if an interval model is given.

The above discussion uses the fact that $\omega \geq 4$. If $\omega \leq 3$, then in Case 3, $|N(u) \cap W| \leq 2$. We can clear u by the following steps. We first place a searcher on one vertex in $N(u) \cap W$, then we move the searcher to u. We then move the searcher back to another vertex (if exists) in $N(u) \cap W$. Hence, ω searchers are sufficient. Thus by Lemma 3.3, we have the following. If $\omega = 2$ then $es(G) = 1$ if and only if G is a path; otherwise, $es(G) = 2$. If $\omega = 3$ then $es(G) = 2$ if and only if $\forall u, v \; |N(u) \cap N(v)| \leq 1$; otherwise, $es(G) = 3$. The conditions can be easily checked, and Algorithm *ESI* can be modified to process interval graphs with any clique size in linear time. Thus we have the following theorem.

Theorem 4.2 *The edge-search number of an interval graph can be computed in linear time.*

5 Node searching on k-starlike graphs

In this section, we discuss the node searching problem on k-starlike graphs. According to [KP85], a node-search strategy can be represented by a sequence of vertex sets $(Y_i) = (Y_0, ..., Y_s)$, where Y_i is the vertex set guarded by searchers at step i. Since recontamination does not occur, thus if v is guarded at step i and cleared at step j then $v \in Y_t$ for $i \leq t \leq j$, and we call such a property the *consecutive property*. Moreover, every edge is cleared at some step i, i.e., for every edge $(u,v) \in E$, there exists some i such that $u, v \in Y_i$. On the other hand, every sequence of vertex sets (Y_i) satisfying the above two properties must correspond to a node-search strategy[KP85].

Let (Y_i) be a node-search strategy. For any maximal clique W of G, there exists a step t such that $W \subseteq Y_t$ [Gu93], i.e., W is cleared at step t. Assume that the clearing order of maximal cliques is given by a permutation $\pi : \{0, \dots, r\} \to \{0, \dots, r\}$, i.e., X_i is cleared before X_j if $\pi(i) < \pi(j)$. Then, it is possible to define a node-search strategy (Y_i) corresponding to such a clearing order as: $Y_0 = X_{\pi^{-1}(0)}$, $Y_r = X_{\pi^{-1}(r)}$, $Y_i = (Y_{i-1} \cap X_0) \cup X_{\pi^{-1}(i)}$ for $0 < i \leq \pi(0)$ and $Y_i = (Y_{i+1} \cap X_0) \cup X_{\pi^{-1}(i)}$ for $\pi(0) \leq i < r$. Such a node-search strategy is called *normalized*. In [Gu93], Gustedt implicitly proved that every starlike graph G has an optimal node-search strategy that is normalized. From now on, we only consider (Y_i) is normalized and assume that G is a k-starlike graph. Note that during a node searching, the number of guarded vertices in central clique is increasing until the whole central clique is guarded and then decreasing. Since each node-search strategy (Y_i) uses $\max_i |Y_i|$ searchers, hence we have the following lemma.

Lemma 5.1 *If $G = (V, E)$ is a k-starlike graph with the maximal cliques $\{X_0, X_1, \dots, X_r\}$, then $|X_0| \leq ns(G) \leq |X_0| + k$.*

Assume that $ns(G) = |X_0| + s$ for some integer s, $0 \leq s \leq k-1$. Let (Y_i) be an optimal node-search strategy, t_0 be the first step when all vertices in X_0 are guarded (i.e., $X_0 \subseteq Y_{t_0}$ and $X_0 \not\subseteq Y_{t_0 - 1}$), and f_l be the lth cleared vertex in X_0 for $1 \leq l \leq k-s$. We claim that all the i-p-cliques containing f_l with $i \geq s + l$ must be cleared before step t_0. Otherwise, there exists an i-p-clique X such that $f_l \in X, i \geq s+l$, and $X \subset Y_{t_l}$ where $t_l \geq t_0$. According to the consecutive property, we know that at step t_l, at least $|X_0| - (l-1)$ vertices in X_0 are guarded, and we need another $i \geq s + l$ searchers to clear the peripheral vertices in X which contradicts the assumption $ns(G) = |X_0| + s$.

Let Q_i be the set of all i-p-cliques, $F_l = \{f_1, f_2, \dots, f_l\}$, and $Q_i^l = \{X \in Q_i | X \cap F_l \neq \emptyset\}$. According to the above discussion, cliques in $Q_k^{k-s} \cup Q_{k-1}^{k-s-1} \cup \cdots \cup Q_{s+1}^1$ must be cleared before step t_0. Now, for $1 \leq i \leq k-s$, consider the step $t < t_0$ such that $|Y_t \cap X_0| \leq |X_0| - i$ and $|Y_{t+1} \cap X_0| > |X_0| - i$. Since between step $t+1$ and step t_0, at most $s + i - 1$ searchers are available to clear peripheral vertices, hence cliques in $Q_k^{k-s} \cup Q_{k-1}^{k-s-1} \cup \cdots \cup Q_{s+i}^i$ must be cleared before step $t+1$. Let $N_i^{k-s} = (Q_k^{k-s} \cup Q_{k-1}^{k-s-1} \cup \cdots \cup Q_{s+i}^i) \cap X_0$. We have $|N_i^{k-s}| \leq |X_0| - i$, or equivalently $|X_0 \setminus N_i^{k-s}| \geq i$.

Conversely, assume there exist vertices $f_1, \dots, f_{k-s} \in X_0$, such that $|X_0 \setminus N_i^{k-s}| \geq i$, for $1 \leq i \leq k-s$. We consider the following clearing order: Clear the cliques in Q_{s+i}^i from $i = k-s$ down to 1, clear the cliques in Q_i with $i \leq s$, and then clear the cliques in $Q_i \setminus Q_{s+i}^i$ from $i = 1$ to $k-s$. It is easy to argue that the resulting node-search strategy corresponding to this clearing order uses at most $|X_0| + s$ searchers. Hence, we have the following lemma.

Lemma 5.2 *Let G be a k-starlike graph with the maximal cliques $\{X_0, X_1, \ldots, X_r\}$ and let s be a given integer, $0 \leq s \leq k-1$. If $ns(G) \geq |X_0| + s$ then $ns(G)=|X_0| + s$ if and only if $\exists f_1, \ldots, f_{k-s} \in X_0$, such that $|X_0 \setminus N_i^{k-s}| \geq i$, for $1 \leq i \leq k-s$.*

By Lemma 5.2, we have an algorithm to determine $ns(G)$ of a k-starlike graph G. The algorithm iteratively execute the following test from $s = 0$ to $k-1$: Is there any sequence of vertices $f_1, f_2, \ldots, f_{k-s}$ in X_0 such that $|X_0 \setminus N_i^{k-s}| \geq i$, for $1 \leq i \leq k-s$? If any of test conditions holds, then the algorithm terminates with output $ns(G) = |X_0| + s$. Otherwise, the algorithm concludes that $ns(G) = |X_0| + k$.

In each iteration, when a sequence of vertices $f_1, f_2, \ldots, f_{k-s}$ is given, a standard graph traversal algorithm can be used to compute N_i^{k-s}, for $1 \leq i \leq k-s$ in $O(m)$ time if each vertex is marked with a flag indicating whether it is in X_0 and if not, which i-p-clique it is in (such information can be obtained by the linear time k-starlike graphs recognition algorithm described in Section 2). Since the algorithm tries all possible $(k-s)$-sequences of vertices in X_0 to test the condition, hence $O((k-s)! \cdot C_{k-s}^{|X_0|} \cdot m) = O((k-s)! \cdot n^{k-s} \cdot m)$ time is needed in each iteration. If k is a fixed constant, then totally the time bound of the algorithm is $O(mn^k)$. It is obvious that $O(m)$-space is sufficient. Thus we have the following theorem.

Theorem 5.3 *There is an $O(mn^k)$-time and $O(m)$-space algorithm to determine the node-search number of a k-starlike graph.*

The algorithm greatly improves a previous algorithm in [Gu93] which uses $O(n^{2k+1})$ time and $O(n^{2k+1})$ space.

6 Edge searching on k-starlike graphs

In this section we present an algorithm for solving the edge searching problem on k-starlike graphs. The algorithm is an extension of the algorithm for the edge searching problem on split graphs given in Section 3 and the algorithm for the node searching problem on k-starlike graphs given in Section 5. We also derive necessary and sufficient conditions for a k-starlike graph to have a given edge-search number by discussing the first k cleared vertices of the central clique in an optimal edge-search strategy.

Lemma 6.1 *If $G = (V, E)$ is a k-starlike graph with the maximal cliques $\{X_0, X_1, \ldots, X_r\}$ and $k \geq 2$, then $|X_0| \leq es(G) \leq |X_0| + k$.*

Proof. The lower bound is obtained from Lemma 2.3. If we have $|X_0| + k$ searchers, $k \geq 2$, then we can first clear the graph by placing $|X_0|$ searchers on X_0. Next, we clear all the edges in X_0 and then apply the algorithm in Lemma 2.4 to clear all the peripheral cliques one by one. Thus, the upper bound is derived. **Q.E.D.**

Before presenting the necessary and sufficient conditions, we need the following lemma.

Lemma 6.2 *Let $\mathcal{S}$ be an edge-search strategy for graph G and let X be a clique of G, $|X| \geq 4$. Let u be the first vertex in X that is cleared using $\mathcal{S}$. Let $S_{t'}$ be the state in $\mathcal{S}$ where u is cleared. Suppose that there exists a state S_t such that $|X \setminus Y_t| \geq 2$, the vertices in $X \setminus Y_t$ are contaminated at S_t and no vertex in $Y_t \setminus \{u\}$ is cleared between S_t and $S_{t'}$. Then at least $|Y_t \cup X|$ searchers are used in $\mathcal{S}$, or each vertex in $X \setminus \{u\}$ is a stain vertex at $S_{t'}$.*

Proof. Since, at $S_{t'}$, all the vertices in $X \setminus \{u\}$ are guarded, there are at least $|Y_t \cup X| - 1$ searchers on the graph. Suppose that only $|Y_t \cup X| - 1$ searchers are used in $\mathcal{S}$. Let x and y be the second last and the last guarded vertex in X. Let x be guarded at S_{t_x}. At S_{t_x}, all $|Y_t \cup X| - 1$ searchers are on the graph. Since between S_{t_x} and $S_{t'}$ there is no free searcher, the edge (u, x) is cleared by $m^+(u, x)$ to reach S_{t_x} and u is cleared after applying $m^-(u, y)$. Thus, each vertex v in $X \setminus \{u\}$ has at least two contaminated incident edges namely, (v, x) and (v, y). **Q.E.D.**

The argument to derive the conditions that determine whether an i-p-clique, $i \geq 2$, should be cleared earlier than the first cleared central vertex is similar to that of the node search of k-starlike graphs. For the 1-p-cliques, the argument is the same as that for the edge search of the split graphs. Let Q_i, F_l, Q_i^l, and N_i^k be the sets defined as in the Section 5. Since for each central vertex c, there exists a state in an optimal search strategy such that c is guarded and it has a free searcher, any 2-p-clique of size three (i.e., 2-p-cliques has only one central vertex c) can be cleared immediately after the state. Thus, we assume without loss of generality that Q_2 does not contain any 2-p-cliques of size three. Let $P_1(v) = \{w \in X | w \in N(v) \setminus X_0$ and $X \in Q_1\}$. Then we have the following lemmas.

Lemma 6.3 *Let G be a k-starlike graph with maximal cliques $\{X_0, X_1, \ldots, X_r\}$ and $k \geq 2$. Then $es(G) = |X_0|$ if and only if $\exists f_1, f_2, \ldots, f_k \in X_0$ such that the following conditions hold.*

(1) $|X_0 \setminus N_i^k| \geq i$, for $i = 2, \ldots, k$,

(2) $|X_0 \setminus (N_2^k \cup N(P_1^{>2}(f_1, f_2) \setminus (A \cup W))| \geq 2$, and

(3) $|X_0 \setminus (N_2^k \cup N(P_1(f_1) \cup P_1^{>2}(f_2) \setminus (B \cup W))| \geq 1$,

where (i) $A \subseteq P_1^{>2}(f_1, f_2)$ and $|A| \leq 1$, (ii) $W \subseteq P_1^{>2}(f_2) \setminus P_1^{>2}(f_1)$ and $|W| \leq 1$, and (iii) $B \subseteq P_1^{=2}(f_1)$ and $|B| \leq 1$.

Proof. Suppose that $es(G) = |X_0|$. Let us consider an optimal edge-search strategy $\mathcal{S}$. Let f_i, $1 \leq i \leq k$, be the ith cleared vertices of X_0 in $\mathcal{S}$. Let x and y be the second last and the last guarded vertex in X_0 respectively. Let S_{t_1} (respectively, S_{t_2}) be the first state in which x (respectively, y) is guarded. Let S_{t_3} be the first state in which f_1 is cleared.

We claim that all peripheral vertices in $X \in Q_k^k \cup Q_{k-1}^{k-1} \cup \cdots \cup Q_2^2$ are cleared before step t_3. If it is not true, let X be an i-p-clique, $i \geq 2$, which is not cleared before step t_3.

First of all, no vertex in $X \backslash X_0$ is cleared at S_{t_3}. Otherwise, there is at least one vertex in $X \backslash X_0$ is guarded at S_{t_3}. However, since all vertices in $X_0 \backslash \{f_1\}$ are guarded, at S_{t_3}, there is exactly one vertex in $X \backslash X_0$, say u, is guarded, and all peripheral vertices in $X \backslash \{u\}$ are clear.

Vertex u must be a stain vertex in S_{t_1-1} by Lemma 2.2. In order to clear u, S_{t_1} is reached by applying $m^+(u,x)$ to S_{t_1-1}. Thus, by Lemma 6.2 for X_0, either $|X_0|+1$ searchers are used in $\mathcal{S}$ or S_{t_3} is deadlock. Both cases lead to a contradiction. Therefore, no vertex in $X \setminus X_0$ is clear at S_{t_3}.

Next, we prove that $X \cap F_i = \emptyset$. Let v_1, that is cleared at $S_{t'}$, be the first cleared vertex in X. At $S_{t'}$, there are at least $i-1$ searchers on vertices of $X \backslash X_0$. It implies that $f_1, \ldots, f_{i-1}$ are not in X and must have been cleared before $S_{t'}$. If $f_i \in X$, by Lemma 6.2, each uncleared vertex in $X \backslash X_0$ is a stain vertex at $S_{t'}$. Together with that each vertex in $X_0 \setminus \{f_1, \ldots, f_i\}$ is also a stain vertex at $S_{t'}$, we obtain that $S_{t'}$ is a deadlock state. Thus, $X \cap F_i = \emptyset$.

Then, we prove the condition (1). Let t_X denote the step that the first vertex in the peripheral clique $X \in Q_i$ is cleared. For $i \geq 2$, let us consider the clique $X \in Q_k^k \cup Q_{k-1}^{k-1} \cup \cdots \cup Q_i^i$ of the maximum t_X. Let u be the first cleared vertex in X, which has to be a peripheral vertex. At S_{t_X}, all vertices in $X \backslash \{u\}$ are guarded and all the vertices in N_i^k are guarded. Let the set of all guarded vertices in X_0 be L. Then $|X_0 \setminus N_i^k| \geq |X_0 \setminus L| \geq |X \setminus X_0| - 1 \geq i-1$. We claim that $|X_0 \setminus N_i^k| \geq i$. Suppose that $|X_0 \backslash N_i^k| = i-1$. It implies that X is an i-p-clique. By Lemma 6.2 for X, at S_{t_X} every vertex in $X \setminus \{u\}$ is a stain vertex at S_{t_X}. Together with the assumption on the optimal edge-search strategy, every vertex in $X_0 \setminus \{f_1, \ldots, f_{i-1}\}$ is a stain vertex. Thus, S_{t_X} is a deadlock state. It contradicts the assumption of the search strategy. Therefore, we conclude that $|X_0 \setminus N_i^k| \geq i$ for all $i \geq 2$.

Next, we prove that conditions (2) and (3) hold. As the argument in the proof Lemma 3.4, we have the following statements.

1. There is at most one 1-p-clique whose peripheral vertex satisfies the following properties. (1) It is of degree larger than 2. (2) It is adjacent to f_2, but not adjacent to f_1. (3) It is adjacent to either the last guarded vertex in X_0 or the second last guarded vertex in X_0.

2. Except the 1-p-clique in item 1, there is no other 1-p-clique whose peripheral vertex is adjacent to f_2 and the last guarded vertex in X_0.

3. There is at most one 1-p-clique whose peripheral vertex is of degree 2 and is adjacent to f_1 and the last guarded vertex in X_0.

4. There is no 1-p-clique whose peripheral vertex is of degree larger than 2 and is adjacent to f_1 and the last guarded vertex in X_0.

5. Except the 1-p-clique in item 1, there is at most one 1-p-clique whose peripheral vertex satisfies the following properties. (1) It is of degree larger than 2. (2) It is adjacent to either f_1 or f_2. (3) It is adjacent to the second last guarded vertex in X_0, but is not adjacent to the last guarded vertex in X_0.

Together with the argument in proving condition (1), we may obtain conditions (2) and (3).

Conversely, if the three conditions hold, we give the following edge-search strategy to clear G.

1. Clear i-p-cliques in Q_i^i from $i = k$ down to 2. Since there are $|X_0 \setminus N_i^k| \geq i$ free searchers, the i-p-cliques in Q_i^i can be cleared one by one by using the algorithm in Lemma 2.4.

2. Clear 1-p-cliques by the same strategy described in Lemma 3.4.

3. Clear i-p-cliques in $Q_i \setminus Q_i^i$ from $i = 2$ to k. Since the searcher at f_i is free after all 1-p-cliques and all i-p-cliques in Q_i^i are cleared, the i-p-cliques in $Q_i \setminus Q_i^i$ can be cleared one by one by using the algorithm in Lemma 2.4.

It is easy to check that the strategy uses $|X_0|$ searchers. **Q.E.D.**

Lemma 6.4 *Let G be a k-starlike graph with the maximal cliques $\{X_0, X_1, \ldots, X_r\}$ and $k \geq 2$. Assume that $es(G) \geq |X_0| + 1$. Then $es(G)$=$|X_0| + 1$ if and only if $\exists f_1, f_2, \ldots, f_{k-1} \in X_0$, such that the following conditions hold.*

(1) $|X_0 \setminus N_i^{k-1}| \geq i$, for $i = 1, \ldots, k-1$, and

(2) $\exists v \in X_0 \setminus N_1^{k-1}$ such that $|P_1^{>2}(f_1) \cap P_1^{>2}(v)| \leq 2$.

Proof. The proof is similar to that in Lemma 6.3. The only difference is the set of peripheral cliques should be cleared before f_1 is cleared is $Q_k^{k-1} \cup Q_{k-1}^{k-2} \cup \cdots \cup Q_2^1$. Since we have one extra searcher, thus for each $(i+1)$-p-clique $X \in Q_k^{k-1} \cup Q_{k-1}^{k-2} \cup \cdots \cup Q_{i+1}^i$, we need additional i free searchers with this extra searcher to clear it. That is $|X_0 \setminus ((Q_k^{k-1} \cup Q_{k-1}^{k-2} \cup \cdots \cup Q_{i+1}^i) \cap X_0)| = |X_0 \setminus N_i^{k-1}| \geq i$ for $i = 1, ..., k-1$. Together with the argument in the proof of Lemma 3.5 on the 1-p-cliques, the condition (2) holds. **Q.E.D.**

Lemma 6.5 *Let G be a k-starlike graph, $k \geq 3$, with the maximal cliques $\{X_0, X_1, \ldots, X_r\}$ and let s be an integer with $2 \leq s \leq k-1$. Assume that $es(G) \geq |X_0|+s$. Then $es(G)$=$|X_0|+s$ if and only if $\exists f_1, \ldots, f_{k-s} \in X_0$, such that $|X_0 \setminus N_i^{k-s}| \geq i$, for $1 \leq i \leq k-s$.*

Proof. The proof is also similar to that of Lemma 6.3. Since we have $|X_0|+s$ searchers with $s \geq 2$, all i-p-cliques with $i \leq s$ can be cleared one by one immediately after the state that all central vertices are guarded. Thus, only i-p-cliques with $i > s$ need to be handled. For the set of all peripheral cliques should be cleared before f_1 is cleared is $Q_k^{k-s} \cup Q_{k-1}^{k-s-1} \cup \cdots \cup Q_{s+1}^1$. By the similar argument described in the proof of Lemma 6.3, our conditions must hold. **Q.E.D.**

By checking the conditions in Lemmas 6.3, 6.4 and 6.5, we have an algorithm to determine $es(G)$ of a k-starlike graph G. The algorithm determines $es(G)$ by checking whether there exists a sequence of k vertices $f_1, f_2, ..., f_k \in X_0$ such that the conditions of Lemma 6.3 hold. If it is not true, then $es(G) \geq |X_0| + 1$. We check the conditions of Lemma 6.4. If it also fails, then we check the conditions in Lemma 6.5 until we can find an s ($\leq k-1$) such that $|X_0 \setminus N_i^{k-s}| \geq i$ holds, for $1 \leq i \leq k-s$. If such an s is found, then $es(G) = |X_0| + s$. Otherwise, $es(G) = |X_0| + k$.

The time complexity depends on that of checking the conditions of Lemmas 6.3, 6.4 and 6.5 for each sequence $f_1, f_2, ..., f_k$. The bottleneck is on the checking of conditions of Lemma 6.3. By using the similar technique in Section 5 for checking the Lemma 5.2, N_i^k can be found in $O(m+n)$ time. The sets A, B and W in conditions (2) and (3) of Lemma 6.3 can be found by the same technique used in Section 3 for checking the conditions of Lemma 3.4. Since there are $O(n^k)$ candidate sequences, the total time used is $O(mn^k)$. Thus we have the following theorem.

Theorem 6.6 *There is an $O(mn^k)$-time and $O(m)$-space algorithm to determine the edge-search number of a k-starlike graph for $k \geq 2$.*

A k-starlike graph is called *pure* k-starlike graph if all of its peripheral cliques are k-p-cliques.

Lemma 6.7 *For any pure k-starlike graph G with $k \geq 3$, $es(G) = ns(G)$.*

Proof. Since $k \geq 3$, there is no 1-p-cliques and 2-p-cliques in G. Thus combine the Lemmas 6.3, 6.4 and 6.5, we obtain that for an integer s, $0 \leq s \leq k-1$, if $es(G) \geq |X_0| + s$ then $es(G) = |X_0| + s$ if and only if $\exists f_1, \ldots, f_{k-s} \in X_0$, such that $|X_0 \setminus N_i^{k-s}| \geq i$, for $1 \leq i \leq k-s$. Compared with Lemma 5.2, the necessary and sufficient conditions are the same for the node search number and the edge search number on pure k-starlike graphs. Thus we have $es(G) = ns(G)$ for a pure k-starlike graph G. **Q.E.D.**

Gustedt [Gu93] proved the pathwidth problem is NP-complete on chordal graphs by reducing the vertex separator problem on an arbitrary graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ to the pathwidth problem on a pure n-starlike graph H with $|V(H)| = mn$. Using the above observation and his result, we have the following theorem.

Theorem 6.8 *The edge searching problem is NP-complete on chordal graphs.*

7 Conclusion

In this paper we present a uniform approach to solve the graph searching problem on split graphs and k-starlike graphs. That is, we derive the necessary and sufficient conditions for a k-starlike (respectively, split) graph to have a given search number by discussing the intersection relation of peripheral cliques with the first k (respectively, 2) cleared vertices of the central clique in an optimal search strategy. Since our algorithms for computing $es(G)$ and $ns(G)$ are constructive, the algorithms can be slightly modified to construct the corresponding search strategies in the same time bound.

Up-to-date, there are very few classes of graphs for which the search number can be computed in polynomial time. It would be interesting to know whether some graph search numbers, especially the edge-search number, can be determined in polynomial time on other special classes of graphs.

References

[ACP87] S. Arnborg, D.G. Corneil, and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM J. Alg. Disc. Meth.*, **8**(1987), 277-284.

[BK96] H.L. Bodlaender and T. Kloks, Efficient and constructive algorithms for the pathwidth and treewidth of graphs, *J. Algorithms*, **21**(1996), 358-402.

[BKK93] H.L. Bodlaender, T. Kloks, and D. Kratsch, Treewidth and pathwidth of permutation graphs, *20th ICALP*, LNCS **700**(1993), 114-125.

[BL76] K.S. Booth and G.S. Leuker, Testing for consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms, *J. Comp. Syst. Scie.*, **13**(1976), 335-379.

[BM93] H.L. Bodlaender and R.H. Moehring, The pathwidth and treewidth of cographs, *SIAM J. Disc. Math.*, **6**(1993), 181-188.

[BS91] D. Bienstock and P. Seymour, Monotonicity in graph searching, *J. Algorithms*, **12**(1991), 239-245.

[EST94] J.A. Ellis, I.H. Sudborough, and J.S. Turner, The vertex separation and search number of a graph, *Information and Computation*, **113**(1994), 50-79.

[Ga74] F. Gavril, The intersection graphs of subtrees in trees are exactly the chordal graphs, *J. Comb. Theory Ser. B*, **16**(1974), 47- 56.

[Gi64] P.C. Gilmore and A.J. Hoffman, A characterization of comparability graphs and of interval graphs, *Can. J. Math.*, **16**(1964), 539-548.

[Go80] M.C. Golumbic, *Algorithmic graph theory and perfect graphs*, Academic Press, New York,1980.

[Gu93] J. Gustedt, On the pathwidth of chordal graphs, *Discr. Appl. Math.*, **45**(1993), 233-248.

[HS81] P.L. Hammer and B. Simeone, The splittance of a graph, *Combinatorica*, **1**(1981), 275-284.

[Ki92] N.G. Kinnersley, The vertex separation number of a graph equals its path-width, *Inform. Process. Lett.*, **42**(1992), 345-350.

[Kl93] T. Kloks, *Treewidth*, Ph.D. Thesis, Utrecht University, The Netherlands, 1993.

[KBMK93] T. Kloks, H. Bodlaender, H. Muller, and D. Kratsch, Computing treewidth and minimum fill-in: all you need are the minimal separators, *First Annual European Symposium on Algorithm*, LNCS **726**(1993), 260-271.

[KP85] L.M. Kirousis and C.H. Papadimitriou, Interval graph and searching, *Disc. Math.*, **55**(1985), 181-184.

[KP86] L.M. Kirousis and C.H. Papadimitriou, Searching and pebbling, *Theoretical Comput. Scie.*, **47**(1986), 205-218.

[KT92] A. Kornai and Z. Tuza, Narrowness, pathwidth, and their application in natural language processing, *Disc. Appl. Math.*, **36**(1992), 87-92.

[La93] A.S. LaPaugh, Recontamination does not help to search a graph, *J. Assoc. Comput. Mach.*, **40**(1993), 224-245.

[MHGJP88] N. Megiddo, S.L. Hakimi, M.R. Garey, D.S. Johnson, and C.H. Papadimitriou, The complexity of searching a graph, *J. Assoc. Comput. Mach.*, **35**(1988), 18-44.

[Mo90] R.H. Moehring, Graph problems related to gate matrix layout and PLA folding, in: G. Tinnhofer et al., eds., *Computational Graph Theory* (Springer, Wien, 1990), 17-32.

[MS88] B. Monien and I.H. Sudborough, Min cut is NP- complete for edge weighted trees, *Theoretical Comp. Scie.*, **58**(1988), 209-229.

[Pa76] T.D. Parsons, Pursuit-evasion in a graph, in Y. Alavi and D.R. Lick, eds., *Theory and applications of graphs*, Springer-Verlag, New York, 1976, 426-441.

[RS83] N. Robertson and P.D. Seymour, Graph minors I. Excluding a forest, *J. Comb. Theory Ser. B*, **35**(1983), 39-61.

[Sc90] P. Scheffler, A linear algorithm for the pathwidth of trees, in: R. Bodendiek and R. Henn, eds., *Topics in Combinatorics and Graph Theory* (Physica-Verlag, Heidelberg, 1990), 613-620.