Contiguous and Internal Graph Searching

Lali Barrière
Pierre Fraigniaud
Nicola Santoro
Dimitrios M. Thilikos

LSI-02-58-R

Contiguous and Internal Graph Searching

Lali Barrière* Pierre Fraigniaud† Nicola Santoro‡ Dimitrios M. Thilikos§

Abstract

In the *graph searching* problem, we are given a graph whose edges are all "contaminated", and, via a sequence of "steps" using "searchers", we want to obtain a state of the graph in which all edges are simultaneously "clear". A *search strategy* is a sequence of search steps that results in all edges being simultaneously clear. The *search number* $s(G)$ of a graph G is the smallest number of searchers for which a search strategy exists.

A search strategy is *monotone* if no recontamination ever occurs; it is *contiguous* if the set of clear edges always forms a connected subgraph; and it is *internal* if searchers, once placed, can only move along the graph edges (i.e., the removal of searchers and their placement somewhere else is not allowed). Depending on the context, each combination of these characteristics may be desirable. Lapaugh proved that, for any graph G, there exists a monotone search strategy for G using $s(G)$ searchers. Obviously, for any graph G there exists an internal search strategy for G using $s(G)$ searchers, but it is not necessarily monotone. We denote by $is(G)$ (resp. $cs(G)$) the minimum number of searchers for which there exists a monotone internal (resp. contiguous) search strategy in G.

We show that, for any graph G, $s(G) \leq is(G) \leq cs(G) \leq 2\ s(G)$. Each of these inequalities can be strict. The last inequality is tight. We actually prove the stronger result stating that, for any graph G, there exists a *monotone contiguous internal* search strategy for G using at most $2\ s(G)$ searchers. As a consequence, the contiguous search number cs is a 2-approximation of pathwidth. Finally, we show that there is a unique *obstruction* for contiguous search and for monotone internal search in trees, in contrast with standard search which involves exponentially many obstructions, even for trees. We prove this result by giving a complete characterization of those searches in trees.

*Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, Spain. *lali@mat.upc.es*.

†CNRS, Laboratoire de Recherche en Informatique, Université Paris-Sud, France. *http://www.lri.fr/~pierre*.

‡School of Computer Science, Carleton University, Canada. *santoro@scs.carleton.ca*.

§Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Spain. *http://www.lsi.upc.es/~sedthilk*.

1 Introduction

Graph searching refers to a problem that has been throughly and extensively investigated in the literature, and that describes a variety of application scenarios ranging from "decontaminating a set of tunnels" to "capturing an intruder in a network".

Using the original metaphor [8, 28, 29], we are given a graph whose edges are all "contaminated", and a set of "searchers". The goal is to obtain a state of the graph in which all edges are simultaneously "clear". To clear an edge $e = (u, v)$, a searcher must traverse the edge from one end-point u to the other end-point v; a clear edge is preserved from recontamination if either another searcher remains in u, or all other edges incident to u are clear. In other words, a clear edge e is recontaminated if there exists a path between e and a contaminated edge, with no searcher on any node of the path. The basic operations, called *search steps*, are: (1) place a searcher on a node, (2) move a searcher along an edge, (3) remove a searcher from a node.

Graph searching is the problem of developing a *search strategy*, that is a sequence of search steps that results in all edges being simultaneously clear. The main complexity measure is the number of searchers used by the strategy; the smallest number of searchers for which a search strategy exists for a graph G is called the *search number* $s(G)$ of G.

The study of graph searching has both practical and theoretical motivations. In particular, graph searching arises in VLSI design, through its equivalence with the gate matrix layout problem (see, e.g., [11, 13, 22]). It is also related to network security for its relation with the capture of an intruder by software agents (see, e.g., [1, 17, 35]), and protection from mobile eavesdroppers [16]. Moreover, the problem and its variants, i.e., *node-search*, *mixed-search*, *inert-search*, etc., are closely related to standard graph parameters and concepts, including treewidth, cutwidth, pathwidth, and linear-width [2]. For instance, $s(G)$ is equal to the cutwidth of G for all graphs of maximum degree 3 (see [25]). Similarly, the node-search number of a graph is equal to its pathwidth plus one, and also to its vertex separator plus one [18, 19, 20]. The inert-search number is equal to the treewidth plus one [10, 32], and the mixed-search number is equal to the proper pathwidth [38, 39]. For more on graph searching, we refer the reader to, e.g., [9, 12, 14, 15].

Graph searching is a non-trivial interesting and challenging problem; even determining whether $s(G) \leq k$ for arbitrary G and k, is NP-complete [26]. Not surprisingly, the research has focused on restricted classes of graphs (e.g., [19, 25, 27, 33, 34]), and on bounded search numbers (e.g., see [6, 28, 37, 39]). In particular, for any fixed k, the class of graphs that can be cleared with up to k searchers is minor closed. Therefore, there is a finite number of *obstructions* for this class [31]; hence, there is a polynomial-time algorithm for testing whether an arbitrary graph G satisfies $s(G) \leq k$, for a fixed k. Of course, the algorithm requires the knowledge of the whole set of obstructions. Unfortunately, the number of obstructions for search grows super-exponentially with k, even for trees [28, 37]. More precisely, for any k, there are at least $(k!)^2$ obstructions for the class of trees T such that $s(T) \leq k$.

Another interesting line of investigation is the determination of efficient search strategies satisfying additional properties, which are desirable or even necessary for some applications. Three properties are of particular interest: absence of recontamination, connectivity of the cleared area, and restricting searchers to move only along the edges.

A search strategy is *monotone* if no recontamination ever occurs. The importance of monotone searching arises in applications where the cost of clearing an edge by far exceeds the cost of traversing an edge. Hence each edge should be cleared only once. Lapaugh [21] has proved that for every G

there is always a monotone search strategy that uses $s(G)$ searchers; a similar positive result exists also for node-search and mixed-search [2, 3].

A search strategy is *contiguous* if the set of clear edges is always connected. The necessity for contiguity arises e.g., in applications where communication between the searchers can occur only within completely clear areas of the network; hence connectivity is required for their coordination. Safety is another motivation for contiguity, as it would always ensure the presence of secure routes between all the searchers. Notice that connectivity does not imply nor is implied by monotonicity. In fact, most existing monotone solutions, after clearing a connected set X of edges, remove the searchers and place them in another part of the graph, usually disconnected from X. The problem of determining minimal search strategies under the contiguity constraint is still NP-complete in general (it follows from the reduction in [26], as observed in [1]); it has been shown in [1] that minimal contiguous strategies can however be computed in linear time for trees.

The next property is perhaps the more practically relevant. A search strategy is *internal* if, once placed, searchers can only move along the graph edges (i.e., they cannot be removed and placed somewhere else). The removal of a searcher from a node x, and the placement of this searcher in another node y, might be difficult or impossible to implement; in fact, it assumes that a searcher is able to go "out of the system" and to reenter the system elsewhere. This assumption is clearly unrealistic e.g., in the case of software mobile agents; in this case the searchers can only move in the network from site to neighboring site. Actually, it does not hold even in the original setting of a maze of caves [28]. Hence the importance of internal search strategies. Restricting the searchers to move only along the graph edges considerably changes the nature of the problem. For instance, there are trees for which minimal internal search strategies require $\Omega(n \log n)$ moves (i.e., edge traversal) [26], whereas, if the removal of searchers (and their arbitrary placement somewhere else) is allowed, then, for any graph G, there exists a minimal search strategy that requires at most $O(n)$ moves in G [21]. Obviously, for any graph G there exists an internal search strategy for G using $s(G)$ searchers. Interestingly, monotone minimal strategies do not always exist for *internal* searching. This makes internal search a singular point in the zoology of search problems. The natural open problem is thus determining the properties of search strategies which are both monotone *and* internal.

Surprisingly, unlike the case of monotone strategies for which there exist detailed studies and characterizations (e.g., [2, 3, 15, 21, 36]), very little is known about contiguous search strategies and monotone internal strategies. Unfortunately, the existing techniques and results for (the many variants of) the problem not only cannot be employed but do not even provide any direct insight on these two important properties.

In this paper we study the properties of these strategies and their relationship with regard to the minimum number of searchers required for their existence.

Our results. Let us denote by $cs(G)$ (resp., $is(G)$) the minimum number of searchers for which there exists a contiguous (resp., monotone internal) search strategy in G. In this paper, we show that for any graph G,

$$s(G) \leq is(G) \leq cs(G) \leq 2\, s(G). \tag{1}$$

Moreover, we show that each of these inequalities can be strict for infinitely many graphs. The last inequality is actually a corollary of a stronger result. In fact, we prove that, for any graph G, there exists a *monotone contiguous internal* search strategy for G using at most $2\, s(G)$ searchers. Hence, all these desirable properties can be satisfied simultaneously by a single search strategy, with no more than twice the minimum number of searchers required to clear the graph without those properties. This also implies that the contiguous search number is a 2-approximation of pathwidth.

To obtain these results, we extend the notion of crusades defined by Bienstock and Seymour [3], and use it in a novel way; in fact, we employ it not to prove monotonicity, but to transform a contiguous strategy into a monotone internal one with the same number of searchers. We also adapt to contiguous search the techniques used by Ellis, Sudborough and Turner [12] for linking search numbers and vertex separation.

We then prove a strong difference between traditional search and both contiguous and monotone internal searches. In fact, we show that, in trees, there is *only one* obstruction for monotone internal search, as well as for contiguous search. This must be contrasted with the fact that, for traditional search, the number of obstructions in trees is *super-exponential* in the number of searchers [28, 37]. We actually provide a complete characterization of the set of trees that can be cleared by k searchers; this characterization is given both explicitly, in terms of *k-caterpillar* (related to the notion of caterpillar dimension of [24]), and implicitly in terms of minimal forbidden minors (i.e., obstructions). As a consequence, we show that there is an infinite family of trees T for which $cs(T) = 2\, s(T) - 2$, proving tightness of the last inequality in equation (1).

Structure of the paper. The paper is organized as follows. In the next section, we prove that $is(G) \leq cs(G)$ for any graph G. In the same section, we also show that there exists graphs G for which $is(G) < cs(G)$. In Section 3, we prove that, for any graph G, there exists a monotone contiguous internal search strategy for G using at most $2\, s(G)$ searchers. In Section 4, we provide the characterization of the trees that can be cleared by k searchers, proving the uniqueness of the obstruction. Finally, Section 5 contains some concluding remarks and open problems.

2 Contiguous vs. Internal Graph Searching

In this section, we prove that $is(G) \leq cs(G)$ for every graph G. We use a generalization of the concept of *crusade* introduced in the short and elegant proof of Bienstock and Seymour [3] of Lapaugh's Theorem.

For a set X of edges in a graph G, we denote by $\delta(X)$ the set of nodes in G having at least one incident edge in X, and at least one incident edge not in X.

Definition 2.1 [3] *Given a graph $G = (V, E)$, a sequence $(X_0, X_1, \ldots, X_r)$ of subsets of edges is a* crusade *if $X_0 = \emptyset$, $X_r = E$, and $|X_i \setminus X_{i-1}| \leq 1$ for any $1 \leq i \leq r$. The* frontier *of a crusade $(X_0, X_1, \ldots, X_r)$ is $\max_{1 \leq i \leq r} |\delta(X_i)|$. A crusade is* progressive *if $X_0 \subseteq X_1 \subseteq \ldots \subseteq X_r$ and $|X_i \setminus X_{i-1}| = 1$ for $1 \leq i \leq r$.*

We say that a crusade is *connected* if the subgraph induced by X_i is connected for any $1 \leq i \leq r$.

Lemma 2.1 *If $cs(G) \leq k$ then there exists a connected crusade of frontier at most k in G.*

Proof. Given a search strategy S in a graph G, let $C = (X_0, X_1, \ldots, X_r)$ be the sequence of subsets of edges such that $X_0 = \emptyset$, and X_i is the set of clear edges after step i of S. At most one edge is cleared at every step of S, and hence $|X_i \setminus X_{i-1}| \leq 1$; i.e., C is a crusade. If S is a search strategy in G using at most k searchers, then obviously the frontier of C is at most k. Clearly, all X_i's are connected for $1 \leq i \leq r$ by definition of contiguous search. ■

Given a crusade $C = (X_0, X_1, \ldots, X_r)$, we define the *skeleton* $\mathcal{S}$ of C as the directed graph of $r+1$ levels L_i, $i = 0, \ldots, r$ such that L_i consists of as many nodes as the number of connected components of X_i. There are edges only between levels of consecutive indices in $\mathcal{S}$ (i.e., each L_i forms a stable). More precisely, there is an edge from the node $a \in L_i$, representing a connected component A of X_i, to the node $b \in L_{i+1}$, representing a connected component B of X_{i+1}, if and only if:

- either $X_{i+1} \setminus X_i \notin B$ and $B \subseteq A$;
- or $X_{i+1} \setminus X_i \in B$ and one of the (at most two) connected component(s) of $B \setminus (X_{i+1} \setminus X_i)$ is included in A.

Note that the out-degree of a node in $\mathcal{S}$ can be greater than 1 because a connected component of X_i can split in several connected components of X_{i+1} due to recontamination. On the other hand, the in-degree of a node is at most 2 because $|X_{i+1} \setminus X_i| \leq 1$ (i.e., there is at most one new clear edge in X_{i+1}); hence at most two distinct connected components of level i can form a unique component at level $i+1$. More precisely, there is at most one node of in-degree 2 at every level of $\mathcal{S}$, and all the other nodes have in-degree ≤ 1. Note also that all nodes in a skeleton of a progressive crusade have out-degree 1 because $X_i \subseteq X_{i+1}$, and hence a connected component never splits.

We denote by $\Gamma^+(u)$ (resp., $\Gamma^-(u)$) the set of edges in $\mathcal{S}$ out-going from (resp., incoming to) node $u \in \mathcal{S}$. Since a node $u \in \mathcal{S}$ represents a set of edges X in G, by extension we denote by $\delta(u)$ the set of nodes in $\delta(X)$.

We now define the concept of *consistent* crusade.

Definition 2.2 *A crusade C is k-*consistent *if its frontier is at most k, and every node u (resp., edge e) of its skeleton $\mathcal{S}$ can be labeled by a positive integer k_u (resp., k_e) satisfying:*
(1) $k_u \geq |\delta(u)|$ for every $u \in \mathcal{S}$;
(2) $\sum_{u \in L_i} k_u \leq k$ for every level L_i;
(3) $k_u = \sum_{e \in \Gamma^+(u)} k_e = \sum_{e \in \Gamma^-(u)} k_e$.

Intuitively, k_u represents the number of searchers in the connected component represented by u. The labels k_e, $e \in \Gamma^+(u)$, represent how the searchers are distributed among the possibly many connected components resulting from a split of the component represented by u. Condition (1) states that the number of searchers in each component is sufficient to protect the component from recontamination. Condition (2) states that the total number of searchers in the graph at any step of a search strategy cannot exceed k. Condition (3) states that: on one hand, a connected component A of X_i shares its searchers among the connected component(s) resulting from the split of A in X_{i+1}. On the other hand, it also states that the number of searchers in a connected component B of X_{i+1} is equal to the sum of searchers coming from component(s) of X_i whose merging results in B.

Observe that the skeleton $\mathcal{S}$ of a connected crusade C is a path. Labeling every node and edge of $\mathcal{S}$ by k makes it k-consistent. Therefore, a connected crusade of frontier at most k is k-consistent. The main reason for introducing consistent crusades is actually the following lemma.

Lemma 2.2 *If there exists a k-consistent crusade in G, then there exists a progressive k-consistent crusade in G.*

Proof. The proof is inspired by (2.2) in [3]. Among all k-consistent crusades, choose a k-consistent crusade $C = (X_0, X_1, \ldots, X_r)$ which satisfies:

(C1) $\sum_{i=0}^{r}(|\delta(X_i)| + 1)$ is minimum, and

(C2) $\sum_{i=0}^{r} |X_i|$ is minimum subject to (C1).

Let us show that this crusade is progressive.

Claim 2.1 $|X_i \setminus X_{i-1}| = 1$ *for every* $i \geq 1$.

Indeed, by contradiction, let i be such that $|X_i \setminus X_{i-1}| = 0$, i.e., $X_i \subseteq X_{i-1}$. Then

$$C' = (X_0, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_r)$$

is a crusade of frontier $\leq k$. Let us show that C' is k-consistent.

In a skeleton, the out-neighbors of a node u are called the children of u, and the out-neighbors of the children of u are called its grandchildren. We define similarly the notion of parents and grandparents.

The skeleton $\mathcal{S}'$ of C' can be obtained from the skeleton $\mathcal{S}$ of C by removing level i, and connecting every node of L_{i-1} to its grandchildren in $\mathcal{S}$. See Figure 1.

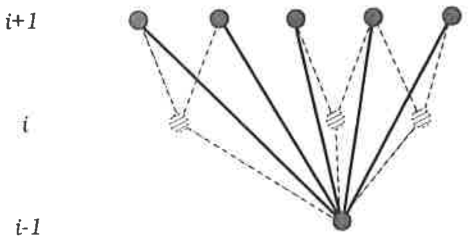

Figure 1: Skeleton $\mathcal{S}'$.

We show that we can label $\mathcal{S}'$ so that the three conditions of Definition 2.2 are satisfied. The node-labeling of $\mathcal{S}'$ is the node-labeling of $\mathcal{S}$. The edge-labeling of $\mathcal{S}'$ is the edge-labeling of $\mathcal{S}$, but between L_{i-1} and L_{i+1}. Edges from L_i to L_{i+1} are labeled as follows. Let $v \in L_{i+1}$. Let u be a grandparent of v in $\mathcal{S}$. There can be at most two distinct paths from u to v in $\mathcal{S}$ because the in-degree of v is at most 2. The edge (u, v) of $\mathcal{S}'$ receives the label of (w, v) of $\mathcal{S}$ if there is a unique path (u, w, v) from u to v in $\mathcal{S}$. It receives the sum of the labels of (w, v) and (w', v) if there are two paths (u, w, v) and (u, w', v) from u to v in $\mathcal{S}$. Since $|X_i \setminus X_{i-1}| = 0$, there is no node of in-degree 2 in L_i of $\mathcal{S}$, and hence this labeling gives k-consistency to $\mathcal{S}'$.

Hence C' is a k-consistent crusade contradicting (C1), and therefore Claim 2.1 holds, i.e., $|X_i \setminus X_{i-1}| = 1$ for every $i \geq 1$.

Next, we show that $X_{i-1} \subseteq X_i$ for every $i \geq 1$.

Claim 2.2 $|\delta(X_{i-1} \cup X_i)| \geq |\delta(X_i)|$ *for every* $i \geq 1$.

Indeed, by contradiction, let

$$C'' = (X_0, X_1, \ldots, X_{i-1}, X_{i-1} \cup X_i, X_{i+1}, \ldots, X_r).$$

C'' is a crusade of frontier $\leq k$. Let us show that it is k-consistent.

The skeleton $\mathcal{S}''$ of C'' can be obtained from the skeleton $\mathcal{S}$ of C by replacing L_i by a copy L'_i of L_{i-1}, and by placing edges between each node and its copy. If the edge $X_i \setminus X_{i-1}$ merges two components of X_{i-1}, then the corresponding two nodes of L'_i are merged into one. Finally, there is an edge from node u' of L'_i to all the grandchildren (in $\mathcal{S}$) of its copy u of L_{i-1}. See Figure 2.

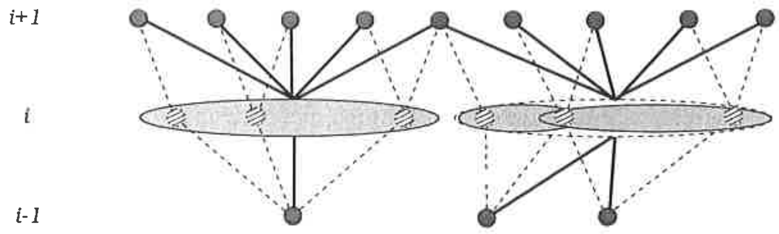

Figure 2: Skeleton $\mathcal{S}''$.

We show that we can label $\mathcal{S}''$ so that the three conditions of Definition 2.2 are satisfied. The node-labeling of $\mathcal{S}''$ is the node-labeling of $\mathcal{S}$ for all nodes of levels $j \neq i$. If a node u of L'_i does not results from the merging of two nodes u' and u'', then u receives the label of its copy in L_{i-1}. Otherwise u receives the sum of the labels of the copies of u' and u'' in L_{i-1}. The edge-labeling of $\mathcal{S}'$ is the edge-labeling of $\mathcal{S}$ except between levels $i-1$, i, and $i+1$. The out-going edge of any node u of L_{i-1} receives label k_u. The setting of the edge-labeling between levels i and $i+1$ is slightly more complex. (Recall that there is at most one node of in-degree 2 at every level.) There is a clear one-to-one correspondence between incoming edges to nodes with in-degree 1 in $\mathcal{S}''$ and $\mathcal{S}$. Thus, the incoming edge of a node at level $i+1$ with in-degree 1 in $\mathcal{S}''$ receives the label of the corresponding edge in $\mathcal{S}$. Let u be a node at level $i+1$ of $\mathcal{S}$, with in-degree 2. If u is still of in-degree 2 in $\mathcal{S}''$ (like in Figure 2), then the two incoming edges of $\mathcal{S}''$ take the same labels as the corresponding edges in $\mathcal{S}$. Otherwise, the unique incoming edge to node u in $\mathcal{S}''$ takes the sum of the labels of the two incoming edges to node u in $\mathcal{S}$. One can easily check that this labeling gives k-consistency to $\mathcal{S}''$.

Therefore C'' is a k-consistent crusade, in contradiction with (C1). Therefore Claim 2.2 holds, i.e., $|\delta(X_{i-1} \cup X_i)| \geq |\delta(X_i)|$.

Now, for any two edge-sets A and B, $|\delta(A \cap B)| + |\delta(A \cup B)| \leq |\delta(A)| + |\delta(B)|$ because every node appearing on the left hand side contributes at least as many times on the right hand side. Thence, we get from Claim 2.2 that $|\delta(X_{i-1} \cap X_i)| \leq |\delta(X_{i-1})|$ for any $i \geq 1$. Let

$$C''' = (X_0, X_1, \ldots, X_{i-2}, X_{i-1} \cap X_i, X_i, \ldots, X_r).$$

C''' is clearly a crusade of frontier at most k.

Claim 2.3 *C''' is k-consistent.*

The skeleton $\mathcal{S}'''$ of C''' can be obtained from the skeleton $\mathcal{S}$ of C by replacing L_{i-1} by a copy L'_{i-1} of L_i, and by placing edges between copies, with the following modification. The node x with in-degree 2 at level i of $\mathcal{S}$ (if any) has two copies in L'_{i-1}. Each copy is connected to u by an edge. We characterize now the edges between level L_{i-2} and L'_{i-1}. Let w be a node of $\mathcal{S}$ at level L_{i-1}. If w has in-degree 1, then the parent of w is connected in $\mathcal{S}'''$ to all the copies of the children of w. If w has in-degree 2, let u and v be the two parents of w in $\mathcal{S}$, and let $x_1, \ldots, x_p$,

$p \geq 0$, be the children of w in $\mathcal{S}$. If $p > 1$, then the connected component w resulting from the merging of two (sub)components (of) u and v is split into pieces. The merging is due to the unique new edge $e_{i-1} = X_{i-1} \setminus X_{i-2}$. Therefore, if w splits into subcomponents, there is at most one subcomponent w' of w that contains e_{i-1}. Any other component is either a subcomponent of u or a subcomponent of v, but not both. Hence, there is an edge from u (resp., v) only to the x_j's which are subcomponents of u (resp., v). See Figure 3.

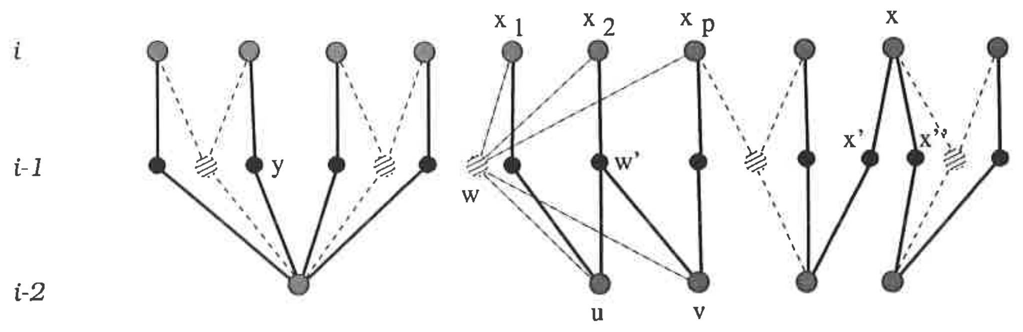

Figure 3: Skeleton $\mathcal{S}'''$.

Knowing the structure of $\mathcal{S}'''$, let us show that we can label $\mathcal{S}'''$ so that the three conditions of Definition 2.2 are satisfied. The node-labeling of $\mathcal{S}'''$ is the node-labeling of $\mathcal{S}$, except for level L'_{i-1}. Similarly, the edge-labeling of $\mathcal{S}'''$ is the edge-labeling of $\mathcal{S}$, except between levels $i-2$, $i-1$, and i. A copy u' of a node $u \in L_i$ with in-degree 1 receives label $k_{u'} = k_u$. Each of the two copies x' and x'' of node x of L'_i with in-degree 2 receive labels that will be specified later. There is a clear one-to-one correspondence between the edges of $\mathcal{S}'''$ and $\mathcal{S}$ incoming to level i. An edge in-coming to L_i in $\mathcal{S}'''$ hence receives the label of its corresponding edge in $\mathcal{S}$. Now, we set up $k_{x'}$ and $k_{x''}$ as the label of the edges (x', x) and (x'', x), respectively. The edge incoming to a node y of in-degree 1 in L'_{i-1} receives the label k_y. The edge $e = (u, w')$ incoming to the node w' of degree 2 in L'_{i-1} receives label $k_u - \sum_{e' \in \Gamma^+(u), e' \neq e} k_{e'}$. We do the same for the edge (v, w'). One can easily check that this labeling gives k-consistency to $\mathcal{S}'''$.

From (C2), Claim 2.3 holds; i.e., $|X_{i-1} \cap X_i| \geq |X_{i-1}|$.

A direct consequence of Claim 2.3 is that $X_{i-1} \subseteq X_i$. Therefore C is a progressive k-consistent crusade, which completes the proof. ■

Lemma 2.3 *Let G be a graph such that every edge has one of its extremities incident to exactly one other edge. If there is a progressive k-consistent crusade in G, then $is(G) \leq k$.*

Proof. Let $C = (X_0, X_1, \ldots, X_r)$ be a progressive k-consistent crusade in G, with skeleton $\mathcal{S}$ labeled as in Definition 2.2. Let $e_i = X_i \setminus X_{i-1} = \{x_i, y_i\}$. We construct an internal search strategy that successively clears the edges $e_1, e_2, \ldots, e_r$. Note that every node of $\mathcal{S}$ has out-degree 1 because C is progressive.

In $\mathcal{S}$, X_1 consists of a unique node u representing $\{e_1\}$. We use k_u searchers to clear e_1. Since, $k_u \geq |\delta(\{e_1\})|$, this number of searchers is sufficient.

Assume now that we have cleared all edges $e_1, \ldots, e_{i-1}$ with an internal strategy. Assume moreover that the number of searchers in each connected component of X_{i-1} is the label of the node corresponding to that component in $\mathcal{S}$.

Let u be a component of level i of $\mathcal{S}$. We consider three cases depending on the in-degree $\deg^-(u)$ of node u.

Case 1: $\deg^-(u) = 0$. Then u consists of a unique edge e_i. It is cleared with k_u searchers.

Case 2: $\deg^-(u) = 2$. Then e_i connects two connected components Y_{i-1} and Z_{i-1} of clear edges. One of the two extremities of e_i, say x_i, is of degree 2 by definition of G. One searcher is staying at x_i to avoid recontamination. The edge e_i is cleared by moving this searcher from x_i to y_i. Hence $k_{Y_{i-1}} + k_{Z_{i-1}}$ searchers are sufficient.

Case 3: $\deg^-(u) = 1$. Then e_i is incident to a unique connected component Y_{i-1} of clear edges. Assume, w.l.o.g., that x_i is the end-point of e_i with degree 2. If $x_i \in \delta(Y_{i-1})$, then e_i is cleared by moving one searcher from x_i to y_i. Thus assume now that $x_i \notin \delta(Y_{i-1})$, which implies $y_i \in \delta(Y_{i-1})$. If $y_i \notin \delta(Y_{i-1} \cup \{e_i\})$, then one searcher can clear e_i by moving from y_i to x_i. Hence assume finally that $y_i \in \delta(Y_{i-1} \cup \{e_i\})$. We get that $|\delta(Y_{i-1} \cup \{e_i\})| = |\delta(Y_{i-1})| + 1$. Now, $k_u \geq |\delta(Y_{i-1} \cup \{e_i\})|$, and hence there is a free searcher in Y_{i-1} that can move to y_i, and clear e_i by moving from y_i to x_i.

Clearly, the search strategy obtained by clearing all edges as explained above is internal. ■

Theorem 2.1 *For every graph G, $is(G) \leq cs(G)$.*

Proof. Let G be any graph with $cs(G) \leq k$. The 1-expansion of G is the graph H obtained from G by replacing every edge e by two consecutive edges e' and e''. We have $cs(H) \leq cs(G)$, by transforming any move along $e \in E(G)$ of a search strategy for G into two moves in H along e' and e''. Therefore, thanks to Lemma 2.1, there exists a connected crusade of frontier $\leq k$ in H. As we noticed before, a connected crusade is k-consistent. Therefore, applying Lemma 2.2, we get that there exists a progressive k-consistent crusade in G. Now, since H is the 1-expansion of G, each of its edges has one of its extremities incident to exactly one other edge. Therefore, by Lemma 2.3, $is(H) \leq k$. We complete the proof by observing that $is(G) \leq is(H)$. Indeed, an internal strategy S_G for G can be obtained from an internal strategy S_H for H as follows. Let e' and e'' be the two incident edges of H resulting from the expansion of an edge e of G. Assume $e' = \{x, y\}$ and $e'' = \{y, z\}$. If one searcher moves from x to y, or from z to y in S_H then this searcher remains in place in S_G. If one searcher moves from y to x (resp., from y to z) in S_H then this searcher moves from z to x (resp., from x to z) in S_G. ■

We conclude the section by proving that the contiguous search number is distinct from the internal and monotone search number, in the sense that there are graphs for which they differ.

Property 2.1 *There exists an infinite family $\mathcal{F}$ of graphs such that $is(G) < cs(G)$ for any $G \in \mathcal{F}$.*

Proof. Let $k \geq 5$, and let G be the graph obtained by "gluing" four copies of the complete graph K_{k-1} to the four corners of a $(2k-1) \times m$ mesh M with $m \geq 2(2k-1)$, m even. More precisely, each corner of M is one of the $k-1$ nodes of a clique of size $k-1$. G has therefore $m(2k-1) + 4(k-2)$ nodes.

Let us first show that $is(G) \leq 2k$. There is an internal and monotone search strategy for G, using $2k$ searchers. Place $2k$ searchers in two corners c and c' at distance $2k-2$ in M, k searchers in each corner. At the corner c, one searcher stays at c, while $k-2$ others move to a distinct neighbor in the associated clique. The remaining free searcher is then used to clear all edges of the clique. The same strategy is applied at c'. Once the two cliques are cleared, the $2k$ searchers are distributed along the shortest path connecting c to c', one searcher per node. The m lines of G are then cleared by moving the $2k-1$ searchers "in parallel", line after line, one free searcher being used to clear all edges of the current line. Once M has been completely cleared, the searchers split into two groups of size k, and each group clears one of the two remaining contaminated cliques.

Let us now show that $cs(G) \geq 2k+1$. Let S be a contiguous search strategy for G. For any $t \geq 1$, let X_t be the set of clear edges in M at step t of S. Let V_t be the set of nodes in M incident to at least one edge of X_t. Let t_0 be such that $|V_{t_0}| = m(2k-1)/2$. Note that S is not necessarily monotone, and the recontamination of an edge at a given step t may imply the simultaneous recontamination of several edges; hence, V_t can be much smaller than V_{t-1}. However, the clearing of an edge can add at most one node at a time in V_t. Hence at least one such a t_0 exists, and, if S is not monotone, may not be unique.

We claim that $|\delta(X_{t_0})| \geq 2k-1$. (Recall that $\delta(X)$ denotes the set of nodes having one incident edge in X, and one incident edge not in X.) Indeed, let I be the smallest rectangle $h \times b$ which contains V_{t_0}, i.e., $|I| = hb \geq m(2k-1)/2$.

If $h < 2k-1$ then $\delta(X_{t_0})$ contains at least b nodes because there are b "unprotected" columns. Thus, since $b > \frac{m}{2} > 2k-1$, we get $|\delta(X_{t_0})| \geq 2k-1$.

If $h = 2k-1$, then V_{t_0} touches two opposite sides of the grid. V_{t_0} can thus be decomposed in $2k-1$ rows of different lengths. If there is no row of length m, then every row needs at least one agent for its protection, and thus $|\delta(X_{t_0})| \geq \#\text{rows} = 2k-1$. If there is a row R of length m, then there is another row R' of length $\ell < m/2$, and thus each of the $m-\ell$ nodes of R' which are not in V_{t_0} requires at least one agent for its protection. Therefore $|\delta(X_{t_0})| \geq m-\ell > \frac{m}{2} \geq 2k-1$.

In all cases, $|\delta(X_{t_0})| \geq 2k-1$. In fact, unless two cliques at distance $2k-2$ are completely decontaminated at time t_0, $|\delta(X_{t_0})| \geq 2k$.

Assume that no two cliques at distance $2k-2$ are completely decontaminated at time t_0. If $|\delta(X_{t_0})| > 2k$ then S uses more than $2k$ searchers, and we are done. Hence assume $|\delta(X_{t_0})| = 2k$. Then the situation is as follows: one searcher is used to protect the mesh from recontamination from a clique, and $2k-1$ searchers form a "front" protecting the mesh from recontamination from the contaminated part. These $2k-1$ searchers alone cannot make much progress towards the other side of the mesh: each searcher can make at most a single move to clear a new edge without recontamination. Therefore, the searchers must backtrack in order to decontaminate two cliques at distance $2k-2$ before trying to clear the mesh. However, this task is impossible if one forces contiguity. Indeed, clearing a clique is fine (using at least k searchers). Then this clique must be protected from recontamination, while a clear path must be created from the clique to the other. However, any such path is contained into a set of clear edges of frontier at least $2k-1$, and two searchers are not enough to clear a clique of size $k-1 \geq 4$. Therefore, S must use more than $2k$ searchers, and thus $cs(G) \geq 2k+1$. (Note that in fact $cs(G) = 2k+1$.) ■

3 Monotone Contiguous Internal Graph Searching

In this section, we prove that, for any graph G, there exists a monotone contiguous internal search strategy using at most $2\, s(G)$ searchers. Obviously, any monotone contiguous search strategy can be transformed into a monotone contiguous *internal* search strategy using the same number of searchers. Indeed, the searchers can move freely in the clear-edges component, to reach any point of the frontier. Therefore, we focus on monotone contiguous search strategies, and characterize those strategies that are minimal.

First let us recall some standard definitions. A *linear layout*, or simply *layout*, of an n-node graph $G = (V, E)$ is a one-to-one mapping $L : V \to \{1, \ldots, n\}$. For $G' = (V', E')$, we denote by $G' \subseteq G$ the fact that G' is a subgraph of G, i.e., $V' \subseteq V$, $E' \subseteq (V' \times V') \cap E$. For any layout L of $G' \subseteq G$, and $1 \leq i \leq |V'|$, let

$$\mathrm{vs}_L(i) = \{x \in V' \ / \ \begin{array}{l} L(x) \leq i, \text{and there exists } y \in V \text{ such that:} \\ \text{either } \{x,y\} \in E' \text{ and } L(y) > i, \\ \text{or } \{x,y\} \in E \setminus E'\} \end{array} \tag{2}$$

The *vertex separation* of $G' \subseteq G$ with respect to L is defined by:

$$\mathrm{vs}_L(G', G) = \max\{|\mathrm{vs}_L(i)|, 1 \leq i \leq |V'|\},$$

and the vertex separation of $G' \subseteq G$ is defined by:

$$\mathrm{vs}(G', G) = \min\{\mathrm{vs}_L(G', G), L \text{ layout of } G' \subseteq G\}. \tag{3}$$

A search strategy for $G' \subseteq G$ is a search strategy which results in all edges of G' simultaneously clear, and preserved from recontamination from edges in $G \setminus G'$. The search number of $G' \subseteq G$ is the minimum number of searchers required for a search strategy in G'. It is denoted by $s(G', G)$.

Let the 2-expansion of G be the graph H formed by replacing each edge $\{x, y\}$ of G by two new vertices, say a and b, and edges $\{x, a\}$, $\{a, b\}$, and $\{b, y\}$. Nodes of the 2-expansion H of a graph G are thus either *original* (i.e., nodes also in G) or *added* (i.e., nodes that have been placed on the edges of G). Let x and y be any pair of original nodes in H. Suppose $L(x) < L(y)$ in some layout L of H. Let a and b be the two nodes added on the edge $\{x, y\}$, where a is adjacent to x, and b to y. L is *standard* if $L(x) < L(a) = L(b) - 1$.

Lemma 3.1 (Theorem 2.2 and Lemma 2.3 in [12]) *Let $G' \subseteq G$, and H (resp., H') be the 2-expansion of G (resp., G'). The three following properties are equivalent:*
(P1) $s(G', G) \leq k$
(P2) $\mathrm{vs}(H', H) \leq k$
(P3) *There exists a standard layout L of $H' \subseteq H$ with $\mathrm{vs}_L(H', H) \leq k$.*

We now extend the theory of Ellis, Sudborough and Turner [12] so to capture the contiguity of the search strategies. A layout L is *connected* if, for any $i \geq 1$, the subgraph of G' induced by vertices $L^{-1}(1), \ldots, L^{-1}(i)$ is connected[1]. The *contiguous vertex separation* of $G' \subseteq G$ is denoted by $\mathrm{cvs}(G', G)$, and is defined by:

$$\mathrm{cvs}(G', G) = \min\{\mathrm{vs}_L(G', G), L \text{ connected layout of } G' \subseteq G\}. \tag{4}$$

Lemma 3.2 *The minimum number of searchers required to clear a graph $G' \subseteq G$ by a monotone contiguous search strategy (and to protect G' from recontamination via edges in $G \setminus G'$) is equal to the contiguous vertex separation $\mathrm{cvs}(H', H)$ of H' in H, where H (resp., H') is the 2-expansion of G (resp., G').*

Proof. A straightforward adaptation of the proof of Lemma 2.1 in [12] allows to show that the minimum number of searchers required to clear a graph $G' \subseteq G$ by a monotone contiguous

[1] Recall that, for any graph H, a subgraph H' of H is an *induced subgraph* of H if $E(H') = E(H) \cap (V(H') \times V(H'))$.

```
CONTIGUOUS SEARCH($L, H', H$)
    Place cvs($H', H$) searchers at node $L^{-1}(1)$;
    for $i := 2$ to $|H'|$ do
        $x := L^{-1}(i)$;
        if $x$ is an original node then
            /* $x$ has necessarily a neighbor $y$ with $L(y) < L(x)$ */
            /* moreover, $y$ is necessarily an added node */
(A)         move a searcher from each neighbor $y$ of $x$ with $L(y) < L(x)$ to $x$;
        else
            /* $x$ is added, with two neighbors $y$ (added) and $z$ (original); */
            /* at least one of them has a label smaller than $L(x)$ */
            if $L(y) < L(x)$ and $L(z) < L(x)$ then
(B)             move a searcher from $y$ to $x$, and then from $x$ to $z$;
            else /* exactly one node $u \in \{y, z\}$ satisfies $L(u) < L(x)$; */
                if $u$ has no neighbor $v$ with $L(v) > L(x)$ then
(C)                 move a searcher from $u$ to $x$ along the edge $\{u, x\}$;
                else /* node $z$ satisfies $L(z) < L(x)$ */
(D)                 move a free searcher from its current position to $z$, and
                        move that searcher from $z$ to $x$;
```

Figure 4: Search strategy in a 2-expansion graph $H' \subseteq H$ with connected layout L.

search strategy is at least cvs(H', H). The proof of the reciprocal uses arguments similar to the ones employed in the proof of Theorem 2.2 in [12]. We sketch these arguments for the sake of completeness. Assume we are given a connected layout of $H' \subset H$ with vertex separation k. The proof of Lemma 2.3 in [12] can be easily adapted to show that there is a *standard* connected layout L of $H' \subset H$ with vertex separation $\leq k$. Based on this layout, we define the search strategy given in Figure 4. The reasons why this strategy is valid are of same nature as those given in [12] for the validation of the procedure SEARCH2. ■

Theorem 3.1 *For any graph G, there exists a monotone contiguous internal search strategy using at most $2\ s(G)$.*

Proof. Let G be a connected graph, and let $G' \subseteq G$. Recall that all considered graphs are simple and loop-less. We prove by induction on the number of nodes in G' that the minimum number of searchers $\alpha(G', G)$ required to clear $G' \subseteq G$ by a monotone and contiguous strategy satisfies $\alpha(G', G)/s(G', G) \leq 2$. For $G' = G$, we shall hence get the result claimed in Theorem 3.1 because a monotone contiguous search strategy can be easily transformed into an internal monotone contiguous search strategy.

The result trivially holds for $|G'| = 1$. Hence assume that G' has at least two nodes. Let S be a minimal search strategy for G' using $k = s(G', G)$ searchers. Using Lapaugh's theorem, we choose S monotone. We transform S into a monotone and contiguous search strategy S' using $\leq 2k$ searchers. From Lemma 3.1, there exists a standard layout L of the 2-expansion H' of G' (considered as a subgraph of the 2-expansion H of G) such that $\text{vs}_L(H', H) = k$. Let x be the original node in the 2-expansion of G' whose label $L(x)$ is maximum among all original nodes of

H'. Since L is standard, all added nodes with labels larger than $L(x)$, if any, are grouped by pairs. Each pair is formed of two consecutive added nodes corresponding to an edge of G'. Therefore, the graph composed of all nodes of H' with label strictly smaller than $L(x)$ is the 2-expansion H'' of some subgraph $G'' \subset G'$ with $|G''| = |G'| - 1$.

First observe that, for such a subgraph G'', $s(G'', G) \leq s(G', G)$. Indeed, the strategy SEARCH2 described in [12] to prove Lemma 3.1, clears the edges by visiting the nodes in increasing order of their label in a standard layout L (in the same way the algorithm described in Figure 4 does). Therefore, the strategy for G' can be stopped after step $L(x) - 1$, resulting in all edges of G'' cleared.

By induction hypothesis, $\alpha(G'', G) \leq 2 \cdot s(G'', G)$, and hence $\alpha(G'', G) \leq 2k$. From Lemma 3.2, there exists a connected layout L'' of the 2-expansion of G'' whose vertex separation is at most $2k$. We complete L'' into a connected layout L' of the 2-expansion H' of G' as follows. We set $L'(x) = |G'|$, and, for every added nodes a in L with $L(a) > L(x)$, we set $L'(a) = L(a) - 1$. In other words, drawing from left to right the nodes labeled in increasing order in L, L' is obtained by: (1) reordering all nodes to the left of x according to L'', (2) placing all added nodes, initially to the right of x in L, in the same order but directly after the rightmost node of L'', and (3) placing x in the rightmost position.

We prove that $\text{vs}_{L'}(H', H) \leq 2k$. Let $i_0 = L(x)$. For $i < i_0$, $\text{vs}_{L'}(i) = \text{vs}_{L''}(i) \leq 2k$. For $i \geq i_0$, $\text{vs}_{L'}(i) \leq \text{vs}_L(i) + p$ where p is the number of neighbors $a_1, \ldots, a_p$ of x such that $L(a_j) > L(x)$. Recall that all a_j's are added nodes. Let b_j be the node of H' added on the same edge as a_j, $j = 1, \ldots, p$. Let y_j be the neighbor of b_j such that $L(y_j) < L(x)$. We have $y_j \neq y_{j'}$ for $j \neq j'$ because the graph G is simple, and $y_j \neq x$ for every j because G is loop-less. Therefore, $\text{vs}_L(i_0) \geq p$, and thus $p \leq k$. Therefore, $\text{vs}_{L'}(i) \leq \text{vs}_L(i) + k \leq 2k$ for every $i \geq i_0$, and hence $\text{vs}_{L'}(G', G) \leq 2k$.

We then apply Lemma 3.2 to get $\alpha(G', G) \leq 2k$, which completes the induction step, and thence the proof. ■

In the next section, we show that the bound of Theorem 3.1 is essentially tight.

4 Obstruction Set for Trees

In this section, we show that there is a unique obstruction for both the class of trees T such that $cs(T) \leq k$, and the class of trees T such that $is(T) \leq k$. Actually, since, in the case of trees, monotone internal search number and contiguous search number are the same [1], we consider contiguous search only. Our proof is based on the notion of *k-caterpillar* and *spine*. A spine is a path. A 0-caterpillar is also a path, and it is its own spine. For $k > 0$, a tree T is a k-caterpillar with spine P if, for every connected component T' of $T \setminus P$, the two following properties hold: (1) there is a path P' such that T' is a $(k-1)$-caterpillar with spine P', and (2) one of the two extremities of P' is adjacent to P. A 1-caterpillar is hence a subdivision of a caterpillar in the usual sense, i.e., a path $x_1, \ldots, x_k$ with $k_i \geq 0$ paths pending from every x_i.

Notice that any tree is a k-caterpillar for k large enough. The notion of k-caterpillar is related to the notion of *caterpillar dimension* introduced in [24] (see also [23]). One can easily show that the contiguous search number of a tree, starting from node v, is at most the caterpillar dimension of the tree rooted at v plus 1. However, as far as we know, there is no characterization of contiguous search number in terms of caterpillar dimension, whereas we establish an equivalence between contiguous search numbers and k-caterpillars. Indeed, we show that k-caterpillars form the class of trees that can be contiguously cleared with at most $k+1$ searchers.

Given a tree T and two vertices v, w of T, we denotes by T_v the tree T rooted at v, and by $T_v[w]$ the subtree of T_v rooted at w. Recall that the depth of a rooted tree T is the maximum distance from its root to the leaves. We denote by B_k the complete binary tree of depth k, and by D_k the tree obtained by connecting the three roots of three copies of B_{k-1} to a unique new vertex. Finally, we denote by $T_1 \preceq T_2$ the relation "T_1 is a minor of T_2".

We now prove a sequence of preliminary lemmas.

Lemma 4.1 *Any tree T such that $D_k \not\preceq T$ is a $(k-1)$-caterpillar.*

Proof. We start by a preliminary statement. Let T_1 and T_2 be two trees, rooted at x_1 and x_2 respectively. We denote by $T_1 \preceq_{x_2} T_2$ the relation "T_1 is a x_2-rooted minor of T_2", that is node x_1 is either x_2 or the result of contracting a series of edges, some of them containing x_2 as end-point. Now, let T be a tree and v be a vertex of T such that $B_k \not\preceq_v T$, $k \geq 1$. We claim that T is a $(k-1)$-caterpillar and v is an extremity of its spine. The proof of that claim is by induction on k. If $B_1 \not\preceq_v T$ then clearly T is a path with extremity v. If $k > 1$ and there is a vertex v such that $B_k \not\preceq_v T$, then there are two cases. If $B_{k-1} \not\preceq_v T$, then by induction hypothesis, T is a $(k-2)$-caterpillar with v as the first vertex of the spine. If $B_{k-1} \preceq_v T$, then let S be the set of vertices w such that $B_{k-1} \preceq_w T_v[w]$. S is a path starting at v, and all the connected components of $T - S$ are $(k-2)$-caterpillars, in which the corresponding spine starts at the vertex adjacent to one of the vertices of S in T. Indeed, if $z \notin S$ and z is adjacent to $w \in S$, then $T_v[z]$ is one of the connected components of $T - S$ and $B_{k-1} \not\preceq T_v[z]$.

To complete the proof of the lemma, it hence just remains to show that there is a vertex v such that $B_k \not\preceq_v T$. By contradiction, assume that $D_k \not\preceq T$ and for every v vertex of T, $B_k \preceq_v T$. There is a vertex z with two neighbors, z_1 and z_2, such that $B_{k-1} \preceq_{z_1} T_z[z_1]$ and $B_{k-1} \preceq_{z_2} T_z[z_2]$. This implies that, either $B_k \preceq_{z_i} T_z[z_i]$ or $B_k \preceq_z T_{z_i}[z]$. In both cases, we get $D_k \preceq T$, a contradiction. ■

Lemma 4.2 *For any $k \geq 1$, $cs(D_k) \geq k+1$.*

Proof. We prove that, for any contiguous search strategy in D_k, there is a step in which at least $k+1$ searchers are required to avoid recontamination. Let T_1, T_2, and T_3 be the three sub-trees attached to the root of D_k and isomorphic to B_{k-1}. Consider the first step i_1 during which the root of D_k is reached by a searcher. Assume, w.l.o.g., that T_1 is still completely contaminated at step i_1. Let $i_2 > i_1$ be the first step during which a leaf of T_1 is reached by a searcher. The path P from the root r to this leaf, say f, has length k. Moreover, at step i_2, P is cleared but, for every vertex $x \neq f$ of P, there is a path from x to a contaminated leaf, and thus at least one searcher is needed for every x to avoid recontamination. Moreover, there is one additional searcher used to clear f. Hence, at least $k+1$ searchers are required at step i_2. ■

Lemma 4.3 *If T is a k-caterpillar then $cs(T) \leq k+1$.*

Proof. We show that, if T is a k-caterpillar with spine P, then there is a contiguous search strategy using $k+1$ searchers starting at one extremity of P. The proof is by induction. For $k=0$, a 0-caterpillar is a path and hence the result holds trivially. Assume now that every $(k-1)$-caterpillar with spine $P' = \{w_0, \ldots, w_\ell\}$ can be cleared with k searchers, starting at w_0. Let T be a k-caterpillar with spine $P = \{v_0, \ldots, v_m\}$. Let us denote by $w_{i,0} \ldots w_{i,d_i}$ the set of neighbors of v_i not in P. Then, $T_{w_{i,j}}[v_i]$ is a $(k-1)$-caterpillar with path $P_{i,j}$ starting at $w_{i,j}$. The search strategy for T is the following. Start at v_0 with $k+1$ searchers. Every time you reach a new vertex v_i of P, let one searcher at v_i and, for $j = 0, \ldots, d_i$, clear every tree $T_{w_{i,j}}[v_i]$ with the k remaining searchers, using the strategy that starts at w_j (there is one, by induction hypothesis). Then, follow the path to the next contaminated vertex v_{i+1}, with the $k+1$ searchers. ∎

Now we are ready to prove the following Theorem.

Theorem 4.1 *For any tree T, the following three properties are equivalent:*

(1) *T is not a $(k-1)$-caterpillar;*

(2) *$D_k \preceq T$;*

(3) *$cs(T) \geq k+1$.*

Proof. The theorem is a direct consequence of the previous lemmas: Lemma 4.1 proves (1)⇒(2), Lemma 4.2 proves (2)⇒(3), and Lemma 4.3 proves(3) ⇒(1) . ∎

Rephrasing Theorem 4.1, we get:

Corollary 4.1 *For a tree T, $cs(T) \leq k$ if and only if T is a $(k-1)$-caterpillar. Moreover, the set of obstructions of the class of trees T with $cs(T) \leq k$ contains D_k as unique element.*

Another consequence of Theorem 4.1 is that the bound of Theorem 3.1 is essentially tight.

Corollary 4.2 *For any tree T, if $s(T) \geq 2$, then $s(T) \leq cs(T) \leq 2s(T) - 2$. Moreover, for $k \geq 1$, $cs(D_{2k-1}) = 2s(D_{2k-1}) - 2$.*

Proof. Let T be a tree, and assume that $s(T) = j$. Let M_k be any tree obtained from a complete ternary tree of depth k after removing one leaf from every set of three sibling leaves (i.e., nodes at distance k from the root). Parsons [28] has proved that M_k is an obstruction of the class of graphs G with search number $\leq k$. Therefore $M_j \not\preceq T$.

Now, M_k is a subgraph of the graph obtained from D_{2k-2} by contracting every edge connecting a vertex of level $2j-1$ to a vertex of level $2j$, for $0 < j < k-1$. Therefore, for any $k \geq 1$, $M_k \preceq D_{2k-2}$. Thus $D_{2j-2} \not\preceq T$, which implies, by Theorem 4.1, that $cs(T) \leq 2j - 2 = 2s(T) - 2$.

To prove that the bound is tight, let us consider D_{2k-1}. We have $s(D_{2k-1}) \leq cs(D_{2k-1}) = 2k$ and $M_k \preceq D_{2k-1}$, which implies that $s(D_{2k-1}) \geq k+1$. On the other hand, we give a search strategy for D_{2k-1} that uses $k+1$ searchers. The strategy starts by placing a searcher in the root r. Next, it proceeds to clear the edges of the three branches which are isomorphic to B_{2k-2}. It is easy to see that this can be done with k searchers, and the edges connecting r to the three branches need no additional searcher. Therefore $cs(D_{2k-1}) = 2s(D_{2k-1}) - 2$, which completes the proof. ∎

5 Concluding Remarks and Open Problems

The main open problem is whether for any graph with $cs(G) \leq k$ there exists a monotone contiguous search strategy for G using $\leq k$ searchers.

A consequence of our results is that graphs with bounded contiguous search number have also bounded pathwidth (node-search number). This implies the existence, for any k, of a linear time algorithm that checks whether $cs(G) \leq k$ and, in case of a positive answer, outputs the corresponding search strategy.

(This follows from the graph minor theory and the result of bodlaender on a linear time algorithm for treewidth [7].)

As we already mentioned, this algorithm is based on the knowledge of the obstruction set that is unknown for the general (non-acyclic) case. it is an interesting open problem to identify this obstruction for small values of k or to construct the corresponding linear-time algorithm. In this direction, techniques like those used in [5, 6, 39] can be useful.

Our result indicates that contiguous search is a 2-approximation of any of the versions of the classic search and, therefore, also of pathwidth. However, it seems that cs is simpler from a structural point of view; thus, it might be possible to compute cs in polynomial time for special classes of graphs, eventhough the pathwidth problem could be NP-hard for those same classes.

References

[1] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. To appear in 14th ACM Symp. on Parallel Algorithms and Architectures (SPAA '02), Winnipeg, August 10-13, 2002.

[2] D. Bienstock. Graph searching, path-width, tree-width and related problems. *DIMACS Series in Disc. Maths. and Theo. Comp. Sc.*, Vol. 5, 33–49, 1991.

[3] D. Bienstock and P. Seymour. Monotonicity in graph searching. *Journal of Algorithms* 12, 239–245, 1991.

[4] D. Bienstock and M. Langston. Algorithmic implications of the graph minor theorem. *Handbooks in OR & MS*, Vol. 7, Chapter 8, 481–502, Elsevier Science, 1995.

[5] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21:358–402, 1996.

[6] H. L. Bodlaender and D.M. Thilikos. Computing small search numbers in linear time. Technical Report UU-CS-1998-05, Utrecht University, 1998.

[7] H. L. Bodlaender, and D. M. Thilikos. Graphs with branchwidth at most three. *Journal of Algorithms*, 32:167–194, 1999.

[8] R. Breisch. An intuitive approach to speleotopology. *Southwestern Cavers* VI(5):72–78, 1967.

[9] H. Buhrman, M. Franklin, J. Garay, J.-H. Hoepman, J. Tromp, and P. Vitányi. Mutual search. *Journal of the ACM*, 46(4):517–536, 1999.

[10] N. Dendris, L. Kirousis, and D. Thilikos. Fugitive-search games on graphs and related parameters. *Theoretical Computer Science*, 172(1–2):233–254, 1997.

[11] J. Díaz, J. Petit, and M. Serna. A survey on graph layout problems. *ACM Computing Surveys*, to appear.

[12] J. Ellis, H. Sudborough, and J. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113(1):50–79, 1994.

[13] M. Fellows and M. Langston. On search, decision and the efficiency of polynomial time algorithm. In 21st *ACM Symp. on Theory of Computing* (STOC '89), pp. 501-512, 1989.

[14] F. Fomin and P. Golovach. Graph searching and interval completion. *SIAM Journal on Discrete Mathematics*, 13(4):454–464, 2000.

[15] F. Fomin and D.M. Thilikos. On the monotonicity of games generated by symmetric submodular functions. *Discrete Applied Mathematics*, to appear.

[16] M. Franklin, Z. Galil, and M. Yung. Eavesdropping games: a graph theoretic approach to privacy in distributed systems. *Journal of the ACM*, 47(2):225–243, 2000.

[17] S. Hansen and M. Eldredge. Intruder isolation and monitoring. In 1st *USENIX Security Workshop*, pages 63–64, 1988.

[18] N. Kinnersley. The vertex separation number of a graph equals its path-width. *Information Processing Letters*, 42(6):345–350, 1992.

[19] L. Kirousis and C. Papadimitriou. Interval graphs and searching. *Discrete Mathematics*, 55:181–184, 1985.

[20] L. Kirousis and C. Papadimitriou. Searching and pebbling. *Theoretical Computer Science*, 47(2):205–218, 1986.

[21] A. Lapaugh. Recontamination does not help to search a graph. *Journal of the ACM*, 40(2):224–245, 1993.

[22] T. Lengauer. Black-white pebbles and graph separation. *Acta Informatica*, 16(4):465–475, 1981.

[23] N. Linial, A. Magen and M. Saks. Trees and Euclidian metrics. In 30st *ACM Symp. on Theory of Computing* (STOC '98), pages 169–175, 1998.

[24] J. Matousek. On embedding trees into uniformly convex Banach spaces. *Israelian Journal of Mathematics*, 114:221–237, 1999.

[25] F. Makedon and H. Sudborough. Minimizing width in linear layout. In 10th *Int. Colloquium on Automata, Languages, and Programming* (ICALP '83), LNCS 154, Springer-Verlag, 478–490, 1983.

[26] N. Megiddo, S. Hakimi, M. Garey, D. Johnson and C. Papadimitriou. The complexity of searching a graph. *Journal of the ACM*, 35(1):18–44, 1988.

[27] S. Neufeld. A pursuit-evasion problem on a grid. *Information Processing Letters*, 58(1):5–9, 1996.

[28] T. Parsons. Pursuit-evasion in a graph. *Theory and Applications of Graphs*, Lecture Notes in Mathematics, Springer-Verlag, pages 426–441, 1976.

[29] T. Parsons. The search number of a connected graph. In 9th *Southeastern Conference on Combinatorics, Graph Theory and Computing*, Utilitas Mathematica, pages 549–554, 1978.

[30] N. Robertson and P. Seymour. Graph minors XIII. The disjoint path problem. *Journal of Combinatorial Theory, Ser. B*, 63:65–110, 1995.

[31] N. Robertson and P. Seymour. Graph minors — A survey. *Surveys in Combinatorics*, Cambridge University Press, I. Anderson (ed.), pages 153–171, 1985.

[32] P. Seymour and R. Thomas. Graph searching, and a min-max theorem for treewidth. *Jour. Combin. Theory, Ser. B*, 58:239–257, 1993.

[33] K. Skodinis. Computing optimal linear layouts of trees in linear time. In 8th *European Symp. on Algorithms* (ESA '00), Springer, LNCS 1879, pages 403-414, 2000. (To appear in *SIAM J. Computing.*)

[34] J. Smith. Minimal trees of given search number. *Discrete Mathematics*, 66:191–202, 1987.

[35] E. H. Spafford and D. Zamboni. Intrusion detection using autonomous agents. *Computer Networks*, 34(4):547–570, 2000.

[36] Y. Stamatiou and D. Thilikos. Monotonicity and inert fugitive search games. In 6th *Twente Workshop on Graphs and Comb. Opt.*, Elsevier, 1999.

[37] A. Takahashi, S. Ueno, and Y. Kajitani. Minimal acyclic forbidden minors for the family of graphs with bounded path-width. *Discrete Mathematics*, 127(1-3):293–304, 1994.

[38] A. Takahashi, S. Ueno, and Y. Kajitani. Mixed searching and proper-path-width. *Theoretical Computer Science*, 137(2):253–268, 1995.

[39] D. Thilikos. Algorithms and obstructions for linear-width and related search parameters. *Discrete Applied Mathematics* 105, 239–271, 2000.

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Departament de Llenguatges i Sistemes Informàtics

Research Reports 2002

- LSI-02-1-R : Anatomía de los antropónimos españoles.
- LSI-02-2-R : From Ternary Relationship to Relational Tables: A Case Against Common Beliefs.
- LSI-02-3-R : MKtree: Generation and Simulations.
- LSI-02-4-R : A characterization of universal stability for directed graphs in the adversarial queueing model.
- LSI-02-5-R : Transforming N-ary Relationships to Database Schemas: An Old and Forgotten Problem.
- LSI-02-7-R : Study on behavioral impedance for route planning techniques from the pedestrian s perspective: Some Findings and Considerations.
- LSI-02-8-R : PROMO: detection of transcription regulatory elements using species-tailored searches .
- LSI-02-9-R : Genome-wide analysis of the Emigrant family of MITEs: amplification dynamics and evolution of genes in Arabidopsis thaliana.
- LSI-02-10-R : Interval Methods for Constructive Geometric Constraint Solving Problems.
- LSI-02-11-R : On-line Support Vector Machines for Function Approximation.
- LSI-02-12-R : Declarative characterization of a general architecture for constructive geometric constraint solvers.
- LSI-02-13-R : Rendering techniques for multimodal data.
- LSI-02-14-R : Supporting Process Reuse in PROMENADE.
- LSI-02-15-R : Evolving Partitions in Conceptual Schemas in the UML (Extended Version).
- LSI-02-16-R : Islands, coordination and parasitic gaps.
- LSI-02-17-R : Revisiting Decomposition Analysis of Geometric Constraint Graphs.
- LSI-02-18-R : MALLBA: A library of skeletons for combinatorial optimisation.
- LSI-02-19-R : An Efficient Representation for Sparse Graphs.
- LSI-02-20-R : Tree Edit Distance and Common Subtrees.
- LSI-02-21-R : Universal stability of undirectd graphs in the adversarial queueing model..
- LSI-02-22-R : The complexity of restrictive H-coloring.
- LSI-02-23-R : Using the Partial Least Squares (PLS) Method to Establish Critical Success Factor Interdependence in ERP Implementation Projects.
- LSI-02-24-R : Recent results on Parameterized H-Coloring.
- LSI-02-25-R : Revisiting variable radius circles in constructive geometric constraint solving.
- LSI-02-26-R : Fast Connected Component Labeling Algorithm: A non voxel-based approach.
- LSI-02-27-R : Problems and conjectures on parameterized H-coloring.
- LSI-02-28-R : Implementation of the methodology "Automatic Characterization and Interpretation of Conceptual Descriptions in ill-Structured Domains using Numerical Variables" (CIADEC) and application to a practical case.
- LSI-02-29-R : Exponential Speedup of Fixed Parameter Algorithms on $K_{3,3}$-minor-free or K_{5}-minor-free Graphs.
- LSI-02-30-R : A Proposal for Wide-Coverage Spanish Named Entity Recognition.
- LSI-02-31-R : H-colorings of Large Degree Graphs.
- LSI-02-32-R : Extending the Carrel System to mediate in the organ and tissue allocation processes: A first Approach.

- LSI-02-33-R : e-Tools for the disabled and for the new generation of senior citizens. A Position paper.
- LSI-02-34-R : The Synonym Management Process in SAREL.
- LSI-02-35-R : Improving Design and Implementation of OO Container-like Component Libraries.
- LSI-02-36-R : Towards the definition of a quality model for mail servers.
- LSI-02-37-R : Design of a Multimodal Rendering System.
- LSI-02-38-R : Visual Clues in Multimodal Rendering.
- LSI-02-39-R : Bisection of Random Cubic Graphs.
- LSI-02-40-R : 1.5-Approximation for Treewidth of Graphs Excluding a Graph with One Crossing as a Minor.
- LSI-02-41-R : Study on behavioral impedance for route planning techniques from the pedestrian s perspective: Part II - Mathematical approach.
- LSI-02-42-R : Searching the Solution Space in Constructive Geometric Constraint Solving with Genetic Algorithms.
- LSI-02-43-R : Creating Agent Platforms to host Agent-Mediated Services that share resources.
- LSI-02-44-R : Fast approximation schemes for K_{3,3}-minor-free or K_{5}-minor-free graphs.
- LSI-02-45-R : Classes of Term Rewrite Systems with Polynomial Confluence Problems.
- LSI-02-46-R : Aplicación de algoritmos de clustering desarrollados en el entorno FIR a la predicción de la concentración de ozono.
- LSI-02-47-R : Random scaled sector graphs.
- LSI-02-48-R : Using Entropy-Based Local Weighting to Improve Similarity Assessment.
- LSI-02-49-R : Boolean operations for 3D simulation of CNC machining.
- LSI-02-50-R : Automatic Construction of Rules Fuzzy for Modelling and Prediction of the Central Nervous System.
- LSI-02-51-R : Caracterización e Interpretación Automática de Descripciones Conceptuales en Dominios Poco Estructurados usando Variables Numéricas.
- LSI-02-52-R : A Two-tiered Methodology to Extend the UML Metamodel.
- LSI-02-53-R : 3D triangular mesh to represent the Left Ventricle Volume of the Heart.
- LSI-02-54-R : Applying KDSM to an specific domain where very short and repeated serial measures with a blocking factor are present.
- LSI-02-55-R : Derivation of algorithms for cutwidth and related graph layout problems.
- LSI-02-56-R : New upper bounds on the decomposability of planar graphs and fixed parameter algorithms.
- LSI-02-57-R : A Framework Proposal for Monitoring and Evaluating Training in ERP Implementation Project.
- LSI-02-58-R : Contiguous and Internal Graph Searching.
- LSI-02-62-R : Feature Selection Algorithms: A Survey and Experimental Evaluation.

[LSI Research Reports Archive | LSI Home Page | UPC Home Page]

Last updated Sep 17, 2002