Monotony Properties of Connected Visible Graph Searching

Pierre Fraigniaud* and Nicolas Nisse

CNRS
Laboratoire de Recherche en Informatique
Université Paris-Sud
91405 Orsay, France
{*pierre,nisse*}*@lri.fr*

Abstract. Search games are attractive for their correspondence with classical width parameters. For instance, the *invisible* search number (a.k.a. *node* search number) of a graph is equal to its pathwidth plus 1, and the *visible* search number of a graph is equal to its treewidth plus 1. The *connected* variants of these games ask for search strategies that are connected, i.e., at every step of the strategy, the searched part of the graph induces a connected subgraph. We focus on *monotone* search strategies, i.e., strategies for which every node is searched exactly once. It is known that the monotone connected visible search number of an n-node graph is at most $O(\log n)$ times its visible search number. First, we prove that this logarithmic bound is tight. Precisely, we prove that there is an infinite family of graphs for which the ratio monotone connected visible search number over visible search number is $\Omega(\log n)$. Second, we prove that, as opposed to the non-connected variant of visible graph searching, "recontamination helps" for connected visible search. Precisely, we describe an infinite family of graphs for which any monotone connected visible search strategy for any graph in this family requires strictly more searchers than the connected visible search number of the graph.
Keywords: Graph Searching, Treewidth, Pathwidth.

1 Introduction

Introduced in [5, 12], graph searching is a game between two players on a graph: one is playing the *fugitive* while the other is playing the *searchers*. They play alternatively. At each step: a searcher is placed at a node, or a searcher is removed from a node; then the fugitive can move from its current node u to any node v in the graph under the constraint that there is a path from u to v that does not cross any node occupied by a searcher. The fugitive is caught when a searcher is placed at the node it occupies . The goal is to find, for every graph G, the

* Both authors received additional supports from the project "PairAPair" of the ACI Masses de Données, from the project "Fragile" of the ACI Sécurité Informatique, and from the project "Grand Large" of INRIA.

minimum k such that there is a *winning* search strategy with k searchers, i.e., a strategy using k searchers that captures any fugitive in G. This minimum k is called the *search number* of the graph (see [3] for a survey on graph searching).

Two main variants of the game have been considered: visible and invisible search. In visible search [6,14], the fugitive is visible to the searchers, and they can thus adapt their search strategy according to the current position of the fugitive. The corresponding search number is called the *visible search* number, denoted by $\mathtt{vs}$. In invisible search [4], the fugitive is not visible to the searchers, and thus they have to perform a blind strategy to capture the fugitive. The corresponding search number is traditionally named the *node search* number. In this paper however, we call it the *invisible search* number for it measures the ability of a team of searchers to capture an invisible fugitive. The invisible search number is denoted by $\mathtt{is}$.

The importance of the search games comes from the correspondence between search numbers and standard width parameters [13], providing different interpretations of these parameters, and hence different ways of handling them. Precisely, it is known that, for any graph G:

- $\mathtt{is}(G) = \mathtt{pw}(G) + 1$ where $\mathtt{pw}(G)$ denotes the *pathwidth* of G (cf. [7]), and
- $\mathtt{vs}(G) = \mathtt{tw}(G) + 1$ where $\mathtt{tw}(G)$ denotes the *treewidth* of G (cf. [6,14]).

Monotony plays a crucial role in graph searching (cf., [11]). A search strategy is *monotone* if once a node has been cleared (a node is cleared at a step of the strategy if the fugitive cannot access to this node at this step), the fugitive cannot ever have access to this node during the rest of the search. Since a monotone search strategy finds the fugitive in a linear number of steps, it gives a polynomially checkable certificate to the decision problem corresponding to a monotone game. Hence the importance of monotony. Proving that visible and invisible search are both monotone games were two major achievements within the theory of graph searching. Precisely, [4,10] proved that if $\mathtt{is}(G) \leq k$ then there exists a winning monotone invisible search strategy using at most k searchers in G. Similarly, [14] proved that if $\mathtt{vs}(G) \leq k$ then there exists a winning monotone visible search strategy using at most k searchers in G.

Connectedness also plays an important role in graph searching, as far as practical applications are concerned (e.g., network security [1], speleological rescue [5], etc). A search strategy in a graph G is *connected* if, at any step of the strategy, the clear part of the graph (i.e., the part of the graph where the fugitive cannot stand) forms a connected subgraph of G. The minimum k for which there is a winning connected search strategy in G using at most k searchers is called the *connected search* number of G. Considering invisible or visible search defines two parameters denoted by $\mathtt{cis}(G)$ and $\mathtt{cvs}(G)$, respectively. The connectivity constraints generally implies a higher number of searchers for capturing the fugitive. The ratio connected search number over search number can however be bounded. Precisely, it is known (see [8]) that for any n-node graph G, we have

$$\mathtt{cis}(G)/\mathtt{is}(G) \leq \log n + 1 \text{ and } \mathtt{cvs}(G)/\mathtt{vs}(G) \leq \log n + 1. \quad (1)$$

For trees, the bound for invisible search can be improved to $\mathtt{cis}(T)/\mathtt{is}(T) \leq 2$ (cf. [2]), and this bound is tight. For visible search, it trivially holds that $\mathtt{cvs}(T) = \mathtt{vs}(T)$ for any tree T.

As for standard (i.e., non-connected) search, monotony is a crucial property for connected search strategies, and it is natural to ask whether monotony holds for connected search games the same way it holds for standard search games. The answer is known to be no for invisible search. Precisely, [15] proves that there is a graph G such that any monotone connected invisible search strategy for G requires more searchers than $\mathtt{cis}(G)$. The impact of this result is important because it is *a priori* difficult to design non-monotone search strategies, and therefore the connected search problem seems significantly harder than the non-connected one. In particular, it is not known whether the decision problem corresponding to connected search is in NP. The good news though is that [1] proves that monotony holds for trees, i.e., for any tree T there is a winning monotone connected invisible search strategy using $\mathtt{cis}(T)$ searchers.

All these results are summarized in Table 1.

	search in arbitrary graphs	connected search in trees		connected search in arbitrary graphs	
	monotone	monotone	ratio	monotone	ratio
invisible fugitive	yes [4, 10]	yes [1]	≤ 2 [2]	no [15]	$\leq \log n + 1$ [8]
visible fugitive	yes [14]	yes [trivial]	1 [trivial]	no **[this paper]**	$O(\log n)$ [8] $\Omega(\log n)$ **[this paper]**

Table 1. An overview of connected graph searching

Our results First, we prove that the bound on the right hand side of Equation 1 is asymptotically tight when restricted to a monotone search strategy. That is, we prove that there is an infinite family of graphs such that, for any n-node graph G in this family, the number of searchers of any winning monotone connected visible search strategy for G is at least $\Omega(\mathtt{vs}(G) \log n)$.

Second, we prove that, as for the connected invisible search game, the connected visible search game is not monotone. Precisely, we describe an infinite family of graphs with arbitrarily large connected visible search number for which any monotone connected visible search strategy for any graph G in this family requires strictly more than $\mathtt{cvs}(G)$ searchers.

Due to lack of place, most of the proofs appear in [9].

2 The lower bound

It is known (cf., [8]) that for any connected n-node graph G, there exists a winning monotone connected invisible search using at most $\mathtt{tw}(G)(\log n + 1)$

searchers. Thus there exists a winning monotone connected visible search using at most $\mathtt{tw}(G)(\log n+1)$ searchers. Since $\mathtt{vs}(G) = \mathtt{tw}(G)+1$, it follows that there exists a winning monotone connected visible search using at most $\mathtt{vs}(G)(\log n+1)$ searchers. We prove that this bound is asymptotically tight.

Theorem 1. *For any n_0, there is $n \geq n_0$ and an n-node graph G such that any winning monotone connected visible search for G uses at least $\Omega(\mathtt{vs}(G) \cdot \log n)$ searchers.*

Proof. We construct an infinite family of connected graphs such that any winning monotone connected visible search for any n-node graph G in this family uses at least $c\,\mathtt{vs}(G)\,\log n$ searchers for some constant $c > 0$. For this purpose, we construct an infinite family $\{G_i, i \geq 1\}$ of connected graphs as follows.

We define the *scale* of length $k > 0$ to be the graph of $2k$ vertices $u_1, \ldots, u_k$, $v_1, \ldots, v_k$ where the u_i's are called *top* nodes, and the v_i's are called *bottom* nodes . There is an edge between u_i and u_{i+1} for all $i = 1, \ldots, k-1$; there is an edge between v_i and v_{i+1} for all $i = 1, \ldots, k-1$; and there is an edge between u_i and v_j for all i, j such that $|i-j| \leq 1$. The *center* of a scale of even length $2k$ is the subgraph induced by the four nodes $u_k, u_{k+1}, v_k, v_{k+1}$. The *extremities* of a scale of length k are the four nodes u_1, v_1, and u_k, v_k, respectively called the left and right extremities.

G_1 is defined as the scale of length $k = 10$, plus one node r_1 called the *root*, and connected to the two extremities u_1 and u_k of the scale . For any $i \geq 1$, the *base* of G_i is a subgraph of G_i that is a scale of even length, and the *kernel* of G_i is the center of its base. For instance, the base of G_1 is the scale of length 10, and the kernel of G_1 is the set $\{u_5, v_5, u_6, v_6\}$, where v_5 and v_6 are the bottom nodes of the kernel of G_1.

Given G_i for $i \geq 1$, we construct G_{i+1} as follows (cf. Fig. 1). Let S_i be the base of G_i (i.e., a scale of even length $2k$), and let r_i be the root of G_i. First, take a copy H of G_i. Let $u_k, u_{k+1}, v_k, v_{k+1}$ be the four nodes of the kernel of H (i.e., the center of the base of H). This kernel is replaced by a scale of length 6, that is: the edges $\{u_k, u_{k+1}\}$, $\{u_k, v_{k+1}\}$, $\{v_k, v_{k+1}\}$, and $\{v_k, u_{k+1}\}$ are removed, u_k and v_k are identified to the left extremities of the length-6 scale, and u_{k+1} and v_{k+1} are identified to the right extremities of the length-6 scale. This operation results in a scale S_{i+1} of length $2k+4$, that becomes the base of G_{i+1}. Next, we take two copies H_1 and H_2 of G_i, and connect the two copies of r_i to the root of H, that becomes the root r_{i+1} of G_{i+1}. Finally, a complete set of connections are added between the two nodes u_k and v_k of H, and the two bottom nodes of the kernel of H_1, and a complete set of connections are added between the two nodes u_{k+1} and v_{k+1} of H, and the two bottom nodes of the kernel of H_2.

We have $|V(G_{i+1})| = 1 + 2|V(G_i)| + (|V(G_i)| - 1 + 8) = 3|V(G_i)| + 8$. Thus $|V(G_i)| = 25 \cdot 3^{i-1} - 4$.

To summarize, we have the base of G_i consisting of a scale of length $2k$ for $k = 2i+3$, with top nodes $u_1, \ldots, u_{2k}$, and bottom nodes $v_1, \ldots, v_{2k}$. The kernel of G_i is the center $\{u_k, v_k, u_{k+1}, v_{k+1}\}$ of this base. Thus the bottom nodes of this kernel are the two nodes v_k and v_{k+1}. The two nodes u_1 and u_{2k} are the top extremities of the base of G_i.

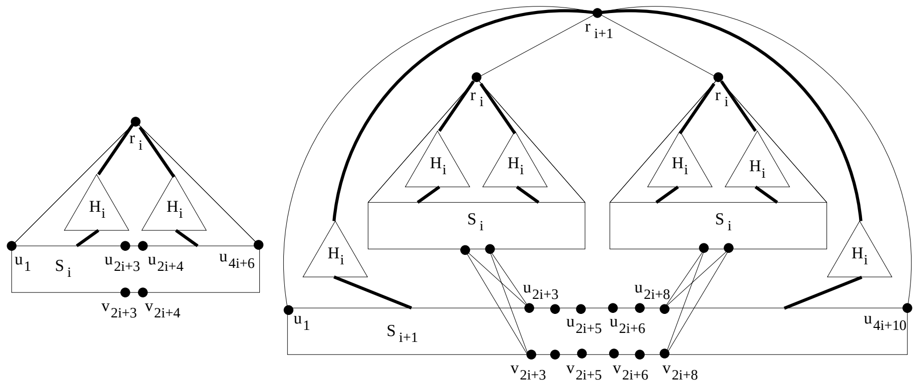

Fig. 1. Recursive construction of G_{i+1} (right) from G_i (left). The dotted lines represent sets of connections.

Claim. For any $i \geq 1$, $\mathtt{tw}(G_i) \leq 4$.

Due to lack place, the proof of this claim is omitted and can be found in [9].

Since for any graph G, $\mathtt{vs}(G) = \mathtt{tw}(G) + 1$, a consequence of Claim 2 is that $\mathtt{vs}(G_i) \leq 5$ for all $i \geq 1$. Before going further in the proof of Theorem 1, we need to present another vision of the graphs G_i. From the definition of G_{i+1}, one can check that it consists of two copies of G_j, for $j = 1, \ldots, i$, connected to a scale of length $2k = 4i + 10$ (cf. Fig. 2). This holds even for $i = 0$ by defining G_0 as the empty graph. More precisely the copies of the G_j's are placed back-to-back in order $G_1, G_2, \ldots, G_i, G_i, \ldots, G_2, G_1$. For every j, the root r_j of any of the two copies of G_j is connected to the root r_{i+1} of G_{i+1}. The two bottom nodes in the kernel of the first copy of G_j are connected to the nodes u_{2j+3} and v_{2j+3} of the base of G_{i+1}, and the two bottom nodes in the kernel of the second copy of G_j are connected to the nodes $u_{2k-(2j+2)}$ and $v_{2k-(2j+2)}$ of the base of G_{i+1}. Finally, the two extremities u_1 and u_{4i+10} of the base scale of G_{i+1} are connected to r_{i+1}. This vision of the graphs G_i's enables us to prove the following.

Claim. For any $i \geq 1$, any winning monotone connected search strategy for G_i whose two first steps consist in placing a searcher at each node v_k and v_{k+1} of the kernel of G_i uses at least $2i + 4$ searchers.

Proof. The proof is by induction on $i \geq 1$. In fact we prove that any monotone connected search strategy starting from v_k and v_{k+1} in G_i has at least $2i + 4$ searchers placed in G_i at the step before it clears the root r_i of G_i. One can easily check that the result holds for G_1, that is any monotone connected search strategy starting from v_5 and v_6 in G_1 has at least 6 searchers placed in G_1 before it clears the root r_1. Let $i \geq 1$ and let us assume that the result holds for any $1 \leq j \leq i$. Let S be a winning monotone connected search strategy for G_{i+1} starting from the two nodes v_k and v_{k+1} of the kernel of G_{i+1}. Consider G_{i+1} as depicted in Fig. 2. To access r_{i+1} from v_k and v_{k+1} in a monotone connected

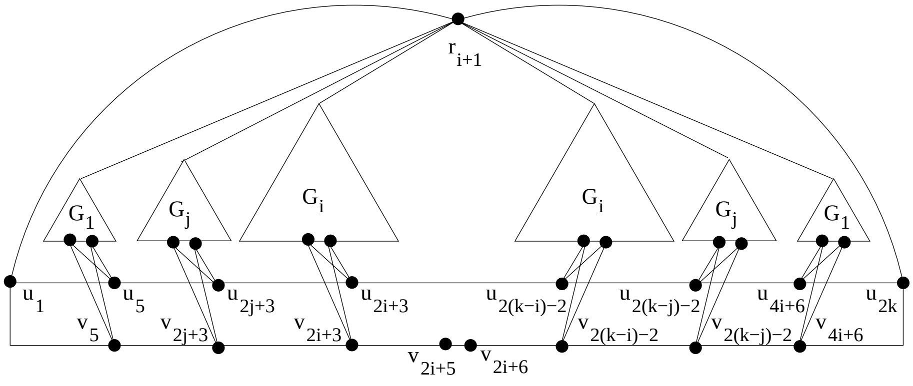

Fig. 2. Alternative definition of G_{i+1}

way, S must clear the root r_j of one of the two copies of some G_j for $1 \leq j \leq i$, or one of the two extremities u_1 or u_{2k} of the base of G_{i+1}. Let R be the set of nodes composed of all the roots of the G_j's composing G_{i+1}, plus the two extremities u_1 and u_{2k}. R contains $2i + 2$ nodes. Let v be the first node in R that is cleared by S. We consider two cases.

The first case assumes that v is one of the two extremities of the base of G_{i+1}. By symmetry of G_{i+1}, one can assume, w.l.o.g., that $v = u_1$. Consider every G_j that is connected to nodes of the base between u_1 and u_k. Recall that the two bottom nodes in the kernel of G_j are connected to the nodes u_{2j+3} and v_{2j+3} of the base. There are two vertex-disjoint paths between the root r_j of the considered G_j to any of the nodes u_{2j+3} and v_{2j+3} of the base. Therefore, if less than two nodes in $V(G_j) \cup \{u_{2j+3}, v_{2j+3}\}$ are occupied by searchers, then one searcher must occupy either u_{2j+3} or v_{2j+3} because otherwise the search will not be connected. Indeed, u_{2j+3} and v_{2j+3} could be contaminated by r_j. Moreover, if one searcher only occupies u_{2j+3} or v_{2j+3}, then another searcher must occupy either u_{2j+4} or v_{2j+4} because otherwise the search will not be connected. As a consequence, for any $1 \leq j \leq i$, at least two nodes of $V(G_j) \cup \{u_{2j+3}, v_{2j+3}, u_{2j+4}, v_{2j+4}\}$ are occupied by searchers. Moreover, two searchers must occupy nodes in $\{u_j, k \leq j \leq 2k\} \cup \{v_j, k \leq j \leq 2k\}$ to avoid recontamination of v_k and v_{k+1} from u_{2k}. Finally, at least four searchers are occupying nodes in $\{u_1, v_1, u_2, v_2, u_3, v_3, u_4, v_4\}$ to connect u_1 with the clear part of G_{i+1}. This yields a total of at least $2i + 6$ searchers in the graph when u_1 is cleared, hence S uses at least $2(i + 1) + 4$ searchers in G_{i+1}.

The second case assumes that the first node $v \in R$ that is cleared by S is the root of some G_j, $1 \leq j \leq i$. We prove that S uses at least $2(i + 1) + 4$ searchers in G_{i+1}. Due to lack of place, the proof of this case is omitted. It can be found in [9]. This completes the induction step, and thus the proof of the claim. ⋄

Let G be connected graph, and let $e = \{u, v\} \in E(G)$. We define the *symmeric* graph of G with respect to e as the graph obtained from two copies of G linked by a set of complete connections between the four nodes resulting from the two copies of $\{u, v\}$. The symmetric of G with respect to $e = \{u, v\}$ is denoted by $G^*_{u,v}$. The K_4 connecting the two copies of G in $G^*_{u,v}$ is called the *center* of $G^*_{u,v}$.

Claim. Let G be a connected graph, and let $\{u, v\} \in E(G)$. Let k be the minimum number of searchers required to clear $G^*_{u,v}$ by a monotone connected visible search strategy. There exists a monotone connected visible search strategy for G using at most k searchers, and whose two first steps consist in placing a searcher at u and a searcher at v.

Due to lack of place, the proof of this claim is omitted and can be found in [9].

For any $i \geq 1$, let $\mathcal{G}_i$ be the symmetric of G_i with respect to $\{v_k, v_{k+1}\}$ where v_k and $v_{k+1}$are the two bottom nodes of the kernel of G_i. We have $|V(\mathcal{G}_i)| = n_i = 2(25 \cdot 3^{i-1} - 4)$. We have $\mathtt{tw}(\mathcal{G}_i) \leq \max\{\mathtt{tw}(G_i), 3\}$ by connecting a bag containing $\{v_k, v_{k+1}\}$ in the tree-decomposition of the first copy of G_i with a bag containing $\{v_k, v_{k+1}\}$ in the tree-decomposition of the second copy of G_i by a path of length two containing a 4-node bag in the middle with two copies of v_k and two copies of v_{k+1}. Hence, from Claim 2, $\mathtt{tw}(\mathcal{G}_i) \leq 4$, and thus $\mathtt{vs}(\mathcal{G}_i) \leq 5$. On the other hand, by combining Claim 2 with Claim 2, we get that any winning monotone connected visible search strategy for $\mathcal{G}_i$ uses at least $2i + 4$ searchers. Therefore, any winning monotone connected visible search strategy for $\mathcal{G}_i$ uses at least $2\log_3(\frac{\frac{n_i}{2}+4}{25}) + 6$ searchers. □

3 Monotony

In this section, we prove that the connected visible search game does not satisfy the monotony property.

Theorem 2. *For any $k \geq 4$, there exists a graph G such that $\mathtt{cvs}(G) = 4k + 1$ and any winning monotone connected visible search strategy uses at least $4k + 2$ searchers.*

Proof. The proof is constructive. For the construction of the graphs mentioned in the statement of the theorem, we reuse the family $\{G_i, i \geq 1\}$ introduced for proving Theorem 1. The intuition of the proof is the following. Consider the graph $I^{(k)}$ depicted in Figure 3. We will show that the symmetric of this graph with respect from $\{u, v\}$ cannot be cleared optimally by a monotone search strategy. In this figure, the graphs E and F are two copies of a graph G_i. Roughly, the placements of these graphs force the strategy to clear them from nodes D and B. We show that it is not possible to do that with the minimal number of searchers in a monotone way.

Claim. There exists a connected visible search strategy for G_i, using at most 5 searchers, and starting from r_i (i.e., the first step of the search consists in placing a searcher at r_i, and the strategy clears the graph by expanding from r_i).

Due to the place limitation, the proof of this claim is omitted and can be found in [9].

Let P_n be the n-node path. Let $P_{k,n}$ be the graph obtained by replacing every vertex of P_n by a complete graph on k vertices, and replacing every edge of P_n by a perfect matching between the complete graphs corresponding to the two extremities of the edge. A graph $P_{k,n}$ is called a *clique-path.*

Claim. For any $n \geq 1$ and any $k \geq 1$:

- There exists a connected visible search strategy for $P_{k,n}$ using at most $k+1$ searchers, and starting from any vertex of the clique corresponding to an extremity of P_n.
- If $n \geq k+1$, then any monotone connected visible search strategy for $P_{k,n}$, using at most k searchers, and starting from any vertex of the clique corresponding to an extremity of P_n cannot clear any vertex of the clique at the other extremity of $P_{k,n}$.

The proof is straightforward and is thus omitted.

For $k \geq 1$, let $I^{(k)}$ be the graph represented in Fig. 3. This representation uses the following coding:

- A black point represents a vertex.
- A circle represents a clique with the indicated number of vertices.
- A thin line between two vertices represents an edge.
- A thin line between a vertex x an a clique represents an edge between x and a vertex of the clique;
- A double line between two cliques represents a perfect matching between them if they are of same size, or between the smallest one and a sub-clique of the largest one if they are of different size.
- a double dotted line between two cliques of same size s represents a path of cliques of size s linked by perfect matchings.
- The graphs K_A, K_B, K_C and K_D are pairwise disjoint k-cliques, all subgraphs of the clique K of size $4k+1$, and extremities of clique-paths.
- The subgraphs E and F are isomorphic to $G_{\lceil 3k/2 \rceil}$ (the marked nodes are the root, and the two bottom nodes of the kernel of $G_{\lceil 3k/2 \rceil}$).

Claim. For any $k \geq 1$, there exists a connected visible search strategy for $I^{(k)}$, starting from u and v, and using at most $4k+1$ searchers.

Proof. The following (non-monotone) strategy uses $4k+1$ searchers. Place searchers at u and v, and use $k+1$ searchers to clear the clique-path leading to A. Let P be a shortest path from A to B going through the central clique K. Place a searcher at every vertex of P, using $2k+3$ searchers (in addition to the k searchers occupying nodes in A). If the fugitive is in the subgraph E, then, from Claim 3, one can use 5 searchers to clear E starting from its root. Thus we assume that the fugitive is not in E. Remove all searchers but $k+1$

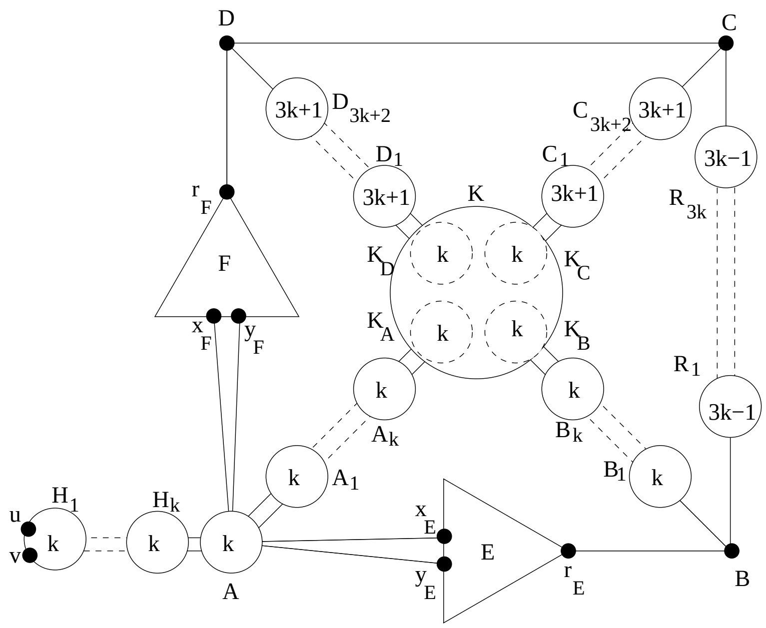

Fig. 3. The graph $I^{(k)}$

searchers occupying A and B, thus E remains isolated. (Note that the strategy is not monotone because of this step). Use the $3k$ remaining searchers to clear the clique-path between B and C (cf. point 1 of Claim 3). After this step, k searchers occupy vertices of A, one searcher occupies B and one searcher occupies C. Place a searcher at D. If the fugitive is in the subgraph F, then, from Claim 3, one can use 5 searchers to clear F starting from its root. Thus we assume that the fugitive is not in F. Use the k searchers at A, plus one extra searcher, to clear the clique-path between A and K_A. At this step, k searchers occupy vertices of K_A, and three searchers occupy B, C, and D. Let us place k searchers at K_D. If the fugitive is in one of the cliques D_i, then remove all searchers but those occupying K_D and D, and use the k searchers at K_D and the $3k$ remaining searchers to clear the clique-path between K_D and D. Thus we assume that the fugitive is not in one of the cliques D_i. Place k searchers at K_C. If the fugitive is in one of the cliques C_i, then remove all searchers but those occupying K_C and C, and use the k searchers at K_C and the $3k$ remaining searchers to clear the clique-path between K_C and C. Thus we assume that the fugitive is not in one of the cliques C_i. Use the searcher at B, and the k remaining searchers to clear the clique-path from B to K_B. At this point, $4k$ searchers occupy vertices of K_A, K_B, K_C and K_D. Use the remaining searcher to clear the last vertex of K. The fugitive is caught, which concludes the strategy. ◇

Note that, since $I^{(k)}$ contains a $4k+1$-clique, the strategy above is optimal.

Claim. For any $k \geq 4$, any winning monotone connected visible search strategy for $I^{(k)}$ starting from u and v uses at least $4k+3$ searchers.

Proof. The proof is inspired from the non-monotony proof for connected invisible search in [15]. In the following, we say that two paths P and P' between a vertex v and a clique are vertex-disjoint if $P \cap P' \in \{v\}$. Let us consider a winning monotone connected visible search strategy S for $I^{(k)}$, starting from u and v.

Let us first assume that the root r_E of E is cleared before vertex B is cleared. Let s be the step at which r_E is cleared in S. Let P be a clear path between u and r_E, and let P' be the subpath of P from u to a vertex in A. Since there are k vertex-disjoint paths between B and P', all passing through the clique K of $I^{(k)}$, k searchers have to guard these paths until step s to avoid recontamination. Moreover, from Claim 2, $G_{\lceil 3k/2 \rceil}$ cannot be cleared by a monotone connected visible search strategy starting from x_E and y_E using less than $3k+4$ searchers. Thus if r_E is cleared before B then S needs at least $4k+4$ searchers. Similarly, one can prove that if r_F is cleared before D then S needs at least $4k+4$ searchers.

Thus, for S to use less searchers, B must be cleared before r_E, and D must be cleared before r_F. Thus, there is a vertex in K_A that is cleared before any of the vertices B, C, and D. Let x be the first vertex of K_A to be cleared by S, say at step s'. (Note that, while none of the vertices B, C and D are cleared, they belong to the same component of the contaminated part, and thus the fact that the fugitive is visible does not help to clear any of these vertices).

Let P_0 be a clear path between u and x at step s'. Let P_1 (resp., P_2) be the subpath of P_0 that goes from u to A (resp., from A_1 to x).

Let us assume that, among B, C, and D, D is the first vertex to be cleared in S. Let $s'' > s'$ be the step when D is cleared. Let P'_1 and P'_2 be two vertex-disjoint paths from r_E to two distinct nodes of P_1. Let P'_3 and P'_4 be two vertex-disjoint paths from r_F to two distinct nodes of P_1 that are as well pairwise distinct from the two extremities of P'_1 and P'_2. Finally, let $P'_5, \dots, P'_{k+4}$ be k vertex-disjoint paths from B to k distinct nodes of P_2. Since $k \geq 4$, these $k+4$ paths can be chosen pairwise vertex-disjoint, and disjoint from any clique D_i. Thus, for any $1 \leq i \leq k+4$, and for any step in $[s', s'']$, there must be a distinct searcher occupying a vertex of P'_i to avoid recontamination of P_0 from r_E, r_F, or B. Point 2 of Claim 3 says that, starting from a vertex of D_1, clearing a vertex of D_{3k+2} in a monotone connected visible way requires at least $3k+2$ searchers. Hence the total number of searchers used by S is at least $4k+6$.

Thus, for S to use less searchers, D should not be the first vertex among B, C, and D to be cleared. Similarly, one can prove that for S to use less searchers, C should not be the first vertex among B, C, and D to be cleared.

Thus, for S to use less searchers, B must be, among B, C, and D, the first node to be cleared by S. Let $s'' > s'$ be the step when B is cleared. At this step, there is a clear path from x to B, through the cliques B_i — recall that we are assuming that B is cleared before r_E. (Note that while C and D are not cleared, both these vertices belong to the same component of the contaminated part, and thus the fact that the fugitive is visible does not help to clear these vertices).

Let P_3 be a clear path from x to B at step s''.

We now consider the two cases depending on whether D is cleared before C, or the other way around.

The first case assumes that D is cleared before C by S. Let $s''' > s''$ be the first step when a searcher is placed at D. Let P'_1 and P'_2 be two vertex-disjoint paths from r_F to two distinct nodes of P_1, and let $P'_3, \ldots, P'_{k+2}$ be k vertex-disjoint paths from C to k disjoint nodes of P_2. Since $k \geq 2$, the $k+2$ paths $P'_1, \ldots, P'_{k+2}$ can be taken pairwise vertex-disjoint, and disjoint from any D_i clique. Thus, for any $1 \leq i \leq k+2$, and for any step in $[s', s''']$, there must be a searcher at a vertex of P'_i to avoid recontamination of P_0 from r_F or C. Point 2 of Claim 3 says that, starting from a vertex of D_1, clearing a vertex of D_{3k+2} in a monotone connected visible way requires at least $3k+2$ searchers. Hence the total number of searchers used by S is at least $4k+4$.

The second case assumes that C is cleared before D by S. Node C can be reached in two different manners: either along the clique-path from C_1 to C_{3k+2}, or along the clique-path from R_1 to R_{3k}. We consider these sub-cases separately. We prove that in both cases, the total number of searchers used by S is at least $4k+3$. Due to the place limitation, the proof of these cases is omitted and can be found in [9].

Therefore, the monotone connected visible strategy S for $I^{(k)}$ uses at least $4k+3$ searchers. ◇

Let $k \geq 4$. Let $G = I^{(k)*}_{u,v}$ be the symmetric of $I^{(k)}$ with respect to the edge $\{u, v\}$. From Claim 3, there exists a connected visible search strategy for $I^{(k)}$, starting from u and v, and using at most $4k+1$ searchers. Therefore $\mathtt{cvs}(G) \leq 4k+1$. On the other hand, Claim 3 states that any winning monotone connected visible search strategy for $I^{(k)}$ starting from u and v uses at least $4k+3$ searchers. By Claim 2, this implies that any winning monotone connected visible search strategy for G uses at least $4k+3$ searchers, that is strictly more than $\mathtt{cvs}(G)$. This completes the proof of the theorem. □

The graphs used in the proof of Theorem 2 have a connected visible search number equal to $4k+1$ for $k \geq 4$, thus at least 17. We can however design examples with smallest search number.

4 Conclusion

In this paper, we first prove that the connectedness requirement for monotone visible search leads to a logarithmic factor in the number of searchers needed. The second result is that the connected visible search is not monotone. A quick glance at Table 1 indicates that our results combined with the previous results in this field let only one problem to be solved, as far as connected search is concerned. Namely: is the bound on the left hand side of Equation 1, i.e., $\mathtt{cis}(G)/\mathtt{is}(G) \leq O(\log n)$, tight? In [2], the authors express their belief that, for any graph G, $\mathtt{cis}(G)/\mathtt{is}(G) \leq 2$. That is, the worst case for connected invisible search is actually reached for trees. Up to now, no one was able to prove or disprove this belief.

We also want to rise the question of minimality for counter examples to monotony of connected search games. Precisely, what is the minimum k such that there is a graph G with $\mathtt{cvs}(G) = k$ for which any winning monotone connected visible search strategy uses more than k searchers. Trivially, $k \geq 3$. Moreover, we prove that $k \leq 4$ [9]. The same question seems far more complex in the context of invisible search (i.e., node search). Indeed, the minimum value that is known for this setting is... $k = 281$ (cf. [15]). Is it possible to design counter examples with smaller connected search numbers?

Finally, what is the complexity of the decision problems "$\mathtt{cis}(G) \leq k$?" and "$\mathtt{cvs}(G) \leq k$?". Both are known to be NP-hard, but are they in NP?

References

1. L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In 14th ACM Symp. on Parallel Algorithms and Architectures (SPAA), pages 200-209, 2002.
2. L. Barrière, P. Fraigniaud, N. Santoro, and D. Thilikos. Connected and Internal Graph Searching. In 29th Workshop on Graph Theoretic Concepts in Computer Science (WG), Springer-Verlag, LNCS 2880, pages 34–45, 2003.
3. D. Bienstock. Graph searching, path-width, tree-width and related problems (a survey). DIMACS Series in Discrete Mathematics and Theoretical Computer Science 5, pages 33-49, 1991.
4. D. Bienstock and P. Seymour. Monotonicity in graph searching. Journal of Algorithms 12, pages 239-245, 1991.
5. R. Breisch. An intuitive approach to speleotopology. Southwestern Cavers VI(5), pages 72-78, 1967.
6. N. D. Dendris, L. M. Kirousis, and D. M. Thilikos. Fugitive search games on graphs and related parameters. Theoretical Computer Science vol. 172, No. 1, pages 233-254, 1997.
7. J. A. Ellis, I.H. Sudborough, J.S. Turner. The Vertex Separation and Search Number of a Graph Information and computation 113, pages 50-79, 1994.
8. P. Fraigniaud and N. Nisse. Connected Treewidth and Connected Graph Searching. Proceedings of Latin American Theoretical Informatics Symposium (LATIN), LNCS 3887, pages 479-490, 2006.
9. P. Fraigniaud and N. Nisse. Monotony Properties of Connected Visible Graph Searching. Technical Report LRI-1456, University Paris-Sud, France, Jul. 2006.
10. A. LaPaugh. Recontamination does not help to search a graph. Journal of the ACM 40(2), pages 224-245, 1993.
11. N. Megiddo, S. Hakimi, M. Garey, D. Johnson and C. Papadimitriou. The complexity of searching a graph. Journal of the ACM 35(1), pages 18-44, 1988.
12. T. Parson. Pursuit-evasion in a graph. Theory and Applications of Graphs, Lecture Notes in Mathematics, Springer-Verlag, pages 426-441, 1976.
13. N. Robertson and P. D. Seymour. Graph minors II, Algorithmic Aspects of Tree-Width. Journal of Algorithms 7, pages 309-322, 1986.
14. P. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width, J. Combin. Theory Ser. B, 58, pages 22-33, 1993.
15. B. Yang, D. Dyer, and B. Alspach. Sweeping Graphs with Large Clique Number. In 5th International Symposium on Algorithms and Computation (ISAAC), Springer, LNCS 3341, pages 908-920, 2004.