Discrete Applied Mathematics 161 (2013) 1950–1958

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Three-fast-searchable graphs

Dariusz Dereniowski [a], Öznur Yaşar Diner [b,*], Danny Dyer [c]

[a] Department of Algorithms and System Modeling, Gdańsk University of Technology, Poland

[b] Department of Information Technology, Kadir Has University, Turkey

[c] Department of Mathematics and Statistics, Memorial University of Newfoundland, Canada

ARTICLE INFO

Article history:
Received 28 February 2011
Received in revised form 12 February 2012
Accepted 4 March 2013
Available online 26 March 2013

Keywords:
Computational complexity
Fast searching
Graph searching

ABSTRACT

In the edge searching problem, searchers move from vertex to vertex in a graph to capture an invisible, fast intruder that may occupy either vertices or edges. Fast searching is a monotonic internal model in which, at every move, a new edge of the graph G must be guaranteed to be free of the intruder. That is, once all searchers are placed the graph G is cleared in exactly $|E(G)|$ moves. Such a restriction obviously necessitates a larger number of searchers. We examine this model, and characterize graphs for which 2 or 3 searchers are sufficient. We prove that the corresponding decision problem is NP-complete.

1. Introduction

Graph searching was first introduced by Breisch in [6], as a problem in spelunking to find an individual lost in a cave. Obviously, in such a situation, the person who is lost may be moving (though not actively avoiding the searchers), and due to the darkness it may be hard to see. Such a person could easily be in the middle of a cave corridor, as opposed to a junction. This problem was first developed mathematically by Parsons in [15], who examined it particularly for trees.

The graph searching (or edge searching) problem is to move agents called *searchers* in such a way as to guarantee the capture of an intruder. Capture occurs when an intruder and a searcher both occupy the same vertex at the same time. Initially, an intruder may be located on any edge or vertex of the graph. Intruders are invisible to the searcher, and as such all edges that may contain an intruder are said to be *contaminated*, or *dirty*. A path that does not contain any searcher is called an *unguarded path*. The intruder can move at any time, and can move from its present location along any unguarded path to any other vertex or edge in the graph. The intruder has full knowledge of the graph and the location of the searchers.

On the other hand, the searchers only stop on vertices. They have full knowledge of the graph, and each others' locations, but not that of any intruders. They move one at a time. In Parsons' original model, only three moves were allowed: a searcher may be placed on any vertex u; a searcher may be removed from any vertex u and a searcher on u may slide along any edge uv. An edge uv becomes *clear* by a sliding move in two ways: There may be (at least) two searchers on u, and one traverses the edge uv while the other remains on u. The second way is when u contains one searcher, denoted by σ, and all edges incident to u, other than uv, are clear. Then that searcher σ traverses uv.

Given a graph G, a sequence of moves using k searchers that ends with all edges of G being clear is called a *k-search strategy* for G; in that case G is called *k-searchable*. The minimum number of searchers needed to clear the edges of G is called the search number of G, denoted by $s(G)$.

In a search strategy, if, after removing a searcher, a clear edge is incident with a vertex that is connected by an unguarded path to a contaminated edge, we say that the clear edge becomes *recontaminated*. A search strategy is said to be *monotonic* if,

* Corresponding author. Tel.: +90 212 5336532; fax: +90 212 533 65 15.
E-mail addresses: deren@eti.pg.gda.pl (D. Dereniowski), oznur.yasar@khas.edu.tr (Ö.Y. Diner), dyer@mun.ca (D. Dyer).

0166-218X/$ – see front matter
http://dx.doi.org/10.1016/j.dam.2013.03.004

after an edge is cleared, searchers must move in a way to never allow recontamination. A search strategy is said to be *internal* if all searchers are first placed in the graph, and then are allowed only to slide along edges, and may never be removed. That is, they may not "jump" from a vertex to a non-adjacent vertex. Using these properties, we analogously define the *monotonic search number*, $ms(G)$, the *internal search number*, $is(G)$, and the *monotonic internal search number*, $mis(G)$. It is straightforward to see that for a connected graph G, $s(G) = is(G)$. It was shown independently in [5,13] that $s(G) = ms(G)$. It is clear that $s(G) \leq mis(G)$, and many examples of graphs exist where the ratio of these parameters may be arbitrarily large [3,18].

In [14], it is shown that computing the search number of trees is polynomial, graphs with search number at most 3 were characterized, and, when combined with the results of [5,13], that the decision problem for search number is NP-complete.

Here we discuss the *fast search* model introduced in [7]. In this model, the intruder behaves as in the standard search model. The searchers have full knowledge of the graph and each others' positions, and move from vertex to vertex. However, after placing all the searchers in the graph, we require that every subsequent move clears an edge and no edge gets recontaminated. In addition the searchers cannot jump from vertex to vertex. Thus, any such strategy must be both monotonic and internal. A strategy that corresponds to such a search is called a *fast search strategy*. The *fast search number*, denoted by $s_f(G)$, is the minimum number of searchers needed for any fast search strategy. If k searchers are enough to fast search G, then we say that G is *k-fast searchable*. The complexity result for fast searching of trees is shown in [7]. It was observed by Yang [17] that the fast searching problem is related with the graph brushing problem [1] and the balanced vertex ordering problem [4]. Other time constrained models are discussed in [2], and a general survey of graph searching is available [9].

This paper develops the fast search problem further. We first characterize those graphs which have fast search number at most 3, and then show that the fast search decision problem is NP-complete through a reduction from a new search model called the *weak searching* problem.

2. Preliminaries

Let G be a connected graph which may have loops or parallel edges. Let S denote a fast search strategy for G. Assume that σ denotes one of the searchers used in S. We say that v is the *start vertex* for σ if σ is initially placed on v according to the strategy S. We say that u is the *end vertex* for σ if σ stops at u (and never moves again). The start and end vertices for σ are denoted by $b(\sigma)$ and $t(\sigma)$, respectively. Moreover, $W(\sigma)$ is the walk traversed by σ during the search of G. We say that the *start vertices* of a search strategy are the vertices of the set $\cup_\sigma b(\sigma)$. Similarly, the *end vertices* of a search strategy are the vertices of the set $\cup_\sigma t(\sigma)$. The *degree* of a vertex u in G, denoted by $\deg_G(u)$, is the number of the endpoints of the edges incident to u in G. This in particular implies that adding a loop to a vertex increases its degree by 2. We call a vertex v even (resp. odd) if it has even (resp. odd) degree. Let V_o be the set of odd vertices in G. Since an odd vertex is a start or an end vertex in a fast search strategy, we have the following lemma.

Lemma 1 *([7]). If V_o is the set of odd degree vertices in a graph G, then $s_f(G) \geq \frac{|V_o|}{2}$.*

Following the development of edge searching we are interested in characterizing graphs where $s_f(G)$ is small.

We define *reduction* as reducing any path with consecutive vertices of degree two to a single edge. Note that reduction does not change the fast search number, and that $s_f(G) = 1$ whenever G can be reduced to a single edge. If a graph G does not contain any vertices of degree 2 then G is a *reduced graph*.

The characterization of 3-searchable graphs is given in [14]. For the characterization of 2-fast searchable graphs G we have the following result.

Theorem 2. *For any reduced graph G, $s_f(G) \leq 2$ if and only if G consists of a path with vertex set $\{v_0, v_1, \ldots, v_n\}$ together with the following conditions:*

1. *For every $i = 0, 1, \ldots, n-1$ there are exactly two parallel edges between each pair of consecutive vertices v_i and v_{i+1}.*
2. *For every $i = 0, 1, \ldots, n$ there may be an arbitrary number of loops attached to each v_i.*
3. *There may be at most two pendant edges attached to v_0 or to v_n.*

Proof. Necessity is obvious. In order to show sufficiency we consider the graphs G for which $s(G) \leq 2$ since $s(G) \leq s_f(G)$ for every graph. It is known [14] that the graphs for which $s(G) \leq 2$ are those that are paths P together with possible pendant edges and loops attached to each vertex. By Lemma 1, we have $|V_\circ| \leq 4$. Let $e = uv_i$ be a pendant edge attached to v_i where v_i is an internal vertex of the path P; hence $i \in \{1, 2, \ldots, n-1\}$. Let $\deg(u) = 1$. Thus u is either a start or an end vertex for a searcher. Assume that u is a start vertex for a searcher. The other case can be shown similarly. This implies that the vertices $v_0, v_1, \ldots, v_{i-1}$ must be cleared by a single searcher. Similarly $v_{i+1}, v_{i+2}, \ldots, v_n$ must be cleared by a single searcher. Thus the graph induced by $v_0, v_1, \ldots, v_{i-1}$ must be a path. Similarly $v_{i+1}, v_{i+2}, \ldots, v_n$ also induce a path. Since G is a reduced graph, $G = K_{1,3}$ or $K_{1,4}$ and $v_0 = v_n = v_i$. This contradicts the assumption that v_i is an internal vertex. Thus no pendant edge is attached to an internal vertex. Hence we have proved Condition 3. Using a similar discussion we conclude that all internal vertices must have even degree, thus Condition 1 and Condition 2 hold. □

3. Characterization of biconnected 3-fast searchable graphs

Here we give the structure of the biconnected graphs that can be cleared with 3 searchers using a fast search strategy. These three searchers are denoted by σ_1, σ_2 and σ_3. First, we give the necessary definitions.

Recall that a biconnected graph is a graph which remains connected when we remove any single vertex. For a graph $G = (V, E)$ and $X \subseteq V$, $G[X]$ denotes the *induced subgraph* that is the subgraph of G with the vertex set X and the edge set $E(G[X]) = \{uv \in E(G) : u, v \in X\}$. For a given pair of vertices u, v, $\mu(u, v)$ is the number of edges between u and v. In this paper, we consider multigraphs and in that case we assume that E is a multiset, i.e. multiple occurrences of an edge with the same endpoints are allowed. Then, $E \setminus \{e\}$ means removing only one occurrence of e from E.

The *contraction* of an edge $uv \in E$ of G is an operation of removing all the edges between u and v, and unifying the vertices u and v, i.e. these vertices are replaced with a new vertex that is adjacent to all the vertices adjacent to u or v in the initial graph preserving the multiplicities of the edges. The *deletion* of an edge $e \in E$ is the operation of removing the edge e. If H is the graph obtained from G by sequence of edge deletions and contractions, then H is called a *minor* of G. The following lemma says that edge searching is closed under taking minors.

Lemma 3 (*[15]*). *If H is a minor of G, then $s(H) \leq s(G)$.*

However, a similar lemma does not hold for fast searching [7] and this forces us to establish different ground rules of characterizational discussions.

A biconnected graph G is *outerplanar* if it has a planar embedding in which the infinite face includes all of its vertices. Consider an outerplanar embedding of a biconnected planar graph G. We say that $uv \in E$ is a *boundary edge* if it is incident to the infinite face. All the remaining edges of G are called *internal edges*. We fix an outerplanar embedding of G, in order boundary edges and internal edges to be well defined.

A *boundary path* in G is a path containing only boundary edges. Note that a boundary path is not necessarily an induced path. Note that the biconnectedness of G implies that the boundary edges form a cycle. If uv and $u'v'$ are different boundary edges then a boundary path P is *connecting* them if it is one of the two boundary paths formed from removing uv and $u'v'$ from the set of all boundary edges. Here the term path means a simple graph, i.e. a path does not contain parallel edges.

A *chord* of a path P in a graph G is an edge $e \in E(G) \setminus E(P)$ between two vertices of P. Two chords e, e' of P are *nested* if the graph $(V(P), E(P) \cup \{e, e'\})$ is contractible to a multigraph with two vertices and three parallel edges between them.

Two edges $e, e' \in E$ are called *opposing poles* of G if the two boundary paths connecting e and e' have no nested chords. An edge $e \in E$ is a pole if there exists $e' \in E$ such that e and e' are opposing poles. If such two edges exist or G consists of a single edge, then we say that G is *bipolar*.

Theorem 4 (*[14]*). *If G is a reduced biconnected multigraph then the search number of G is at most 3 if and only if G is outerplanar and bipolar.*

Since every fast search strategy is also a search strategy, by Theorem 4 we have the following result.

Corollary 5. *If a reduced biconnected graph G is 3-fast searchable then G is bipolar and outerplanar.* □

To be able to conclude that a property shown for the set of start vertices is also true for the set of end vertices in some search strategy, we use the concept of reversibility.

Definition 1 (*[16]*). Given a search strategy S, define the *reverse* of a step $a \in S$, denoted by a^{-1}, as follows:

- If a is 'slide σ_i from v to u' then a^{-1} is 'slide σ_i from u to v'.
- If a is 'remove σ_i from v', then a^{-1} is 'place σ_i on v'.
- If a is 'place σ_i on v', then a^{-1} is 'remove σ_i from v'.

Given a search strategy $S = (a_1, a_2, \ldots, a_l)$ that uses l steps, define the *inverse* of S, denoted by S^{-1} to be $S^{-1} = (a_l^{-1}, a_{l-1}^{-1}, \ldots, a_1^{-1})$.

We say that a search strategy S is *reversible* if S^{-1} is a search strategy. Lemma 3 in [16] proves that monotonic search strategies are reversible. However, their proof can be easily generalized to general search strategies on general graphs. Moreover, if S is a fast search strategy, then an operation of sliding a searcher along an edge e appears in S^{-1} exactly once for each edge e. This gives us the following lemma.

Lemma 6. *If S is a fast search strategy that uses k searchers for a multigraph G, then S^{-1} is a fast search strategy that uses k searchers for G.* □

Lemma 7. *If G is a reduced biconnected 3-fast searchable multigraph then for any fast search strategy with 3 searchers, the set X of start (end) vertices is such that $|X| = 1$, or $|X| = 2$ and the two vertices in X are adjacent.*

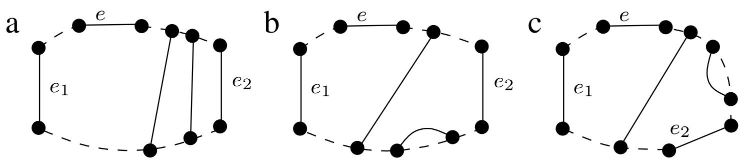

Fig. 1. Nested chords of a boundary path between e and e_1.

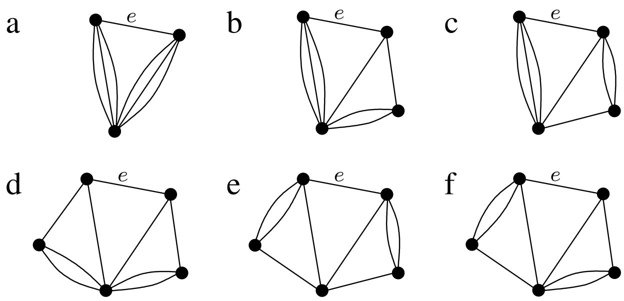

Fig. 2. The graphs H such that G is contractible to H.

Proof. Assume that $S = (s_1, \ldots, s_l)$ is a search strategy for G. Assume for a contradiction that the set of start vertices X is of size 3. When the first move of sliding a searcher σ_i from u to v occurs, then $\deg_G(u) = 1$, because no other searcher is at u. This contradicts the fact that G is biconnected. Hence $|X| \le 2$.

Let the initial placement of the searchers be $u = b(\sigma_1) = b(\sigma_2) \ne b(\sigma_3) = v$ and suppose that there is no edge between u and v. As before, σ_3 cannot change his position until another searcher reaches v. Then, after either of σ_1 or σ_2 move, they are on vertices of degree at least three since G is reduced. Each of those two vertices must be incident to at least two contaminated edges, and hence unable to move. Thus, the edges incident to v can never be clear, a contradiction. Therefore there is an edge between u and v. By Lemma 6, the property also holds for the set of end vertices. □

Lemma 8. *If G is a reduced biconnected outerplanar bipolar 3-searchable multigraph then for each monotonic search strategy using 3 searchers we have $b(\sigma_i) \in e$ for each $i = 1, 2, 3$ and $t(\sigma_i) \in e'$ for each $i = 1, 2, 3$, where e, e' are some opposing poles of G. Furthermore, e is the first cleared edge and e' is the last.*

Proof. By Lemma 7 the set X of start vertices has size at most 2. If all the searchers start at the same vertex u then the first move of a searcher results in a situation when one edge e incident to u is clear and both end vertices of $e = uv$ are occupied by the searchers. If the set of starting vertices contains more than one vertex then by Lemma 7, $u = b(\sigma_1) = b(\sigma_2) \ne b(\sigma_3) = v$ and $uv \in E$. Then, if $e = uv$ is not the first cleared edge, then all searchers get stuck after the first move (as G is a reduced graph).

Thus, we let $e = uv$ be the first cleared edge. We prove that in both cases e is a pole of G.

Fix an outerplanar embedding of G. Assume that e is not a pole. Let e_1, e_2 be two poles of G. We consider two cases and in both of them we obtain a contradiction by proving that 3 searchers are not sufficient to clear G.

Case 1: e is a boundary edge. The edges e and e_1 are not opposing poles. Let P_1, P_2 be the two boundary paths connecting e and e_1 in G. One of these paths contains e_2. Assume without loss of generality that P_2 contains e_2. Note that P_1 cannot have nested chords because then a boundary path connecting e_1 and e_2 would have nested chords. Thus, P_2 has two nested chords and one of these chords has a subpath that contains e_2 between its end vertices.
All possible configurations of chords of P_2 are depicted in Fig. 1.
Similarly, since e and e_2 are not opposing poles and since we consider a planar embedding of G, the pairs of chords related to the pairs e, e_1 and e, e_2 are edge-disjoint. Then every such graph contains one of the graphs shown in Fig. 2 as a minor. Call this minor H.
It is easy to see that it is impossible to search H if the three searchers are initially placed on the vertices incident to e and e is the first cleared edge. By Lemma 3, there exists no 3-search strategy that clears G, such that the first edge cleared is e. Thus, no 3-fast search strategy for G starting by clearing e exists. This leads to a contradiction.

Case 2: e is an internal edge. Let P_1 and P_2 be the two different boundary paths connecting the end vertices of e. Assume, without loss of generality, that $e_i \in E(P_i)$ for $i = 1, 2$. For each $i = 1, 2$, P_i has a chord, because otherwise either G is not a reduced graph or e is a boundary edge. There are two possibilities when we take into consideration the position of the chord of P_i with respect to the pole e_i. Both possibilities in the case of P_2 are depicted in Fig. 3(a). As before, each such graph contains one of the graphs in Fig. 3(b) as a minor H. Then it is easy to see that, by Lemma 3, it is not possible to clear G when all the searchers are initially placed at the vertices of e. The situation for P_1 is analogous.

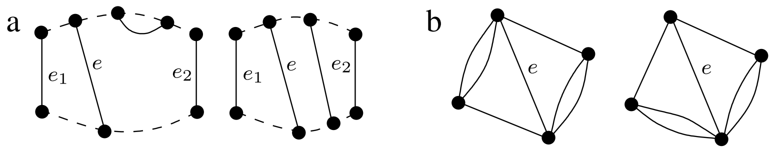

Fig. 3. (a) A graph G' with two possible positions of the chord of P_2; (b) possible graphs H such that G' is contractible to H.

In this way we showed that in each 3-fast search strategy $b(\sigma_i) \in e$ for some pole e of G. Since each search strategy is reversible, we have $t(\sigma_i) \in e'$ for a pole e' of G. It is easy to see that e and e' must be opposing poles. $\square$

We define *starting* and *ending poles* to be a pair of opposing poles in a biconnected bipolar 3-fast searchable graph from which the searchers in a 3-fast search strategy start and end. Thus e and e' from the statement of Lemma 8 is a pair of starting and ending poles.

Lemma 9. *If G is a reduced biconnected 3-fast searchable graph then*

1. *if all vertices of G are even, then G contains a pair of adjacent starting and ending poles;*
2. *if G has an odd vertex, then there are exactly two odd vertices, one of which is a vertex of a starting pole and the other is a vertex of a corresponding ending pole.*

Proof. Let G be a reduced biconnected 3-fast searchable graph, and e and e' be the starting and ending poles of G. By Lemma 1, $|V_o| \leq 6$. On the other hand by Lemma 8, the searchers start and end at the poles, thus $V_o \subseteq e \cup e'$, and thus $|V_o| \leq 4$. Since the number of odd vertices must be even, we must have 0, 2 or 4 odd vertices. If the poles are incident, then $|V_o| \leq 3$, and hence $|V_o| \leq 2$. Assume that the poles are not incident and $|V_o| = 4$. Then, each vertex of each pole must be odd. Lemma 8 implies that each pole contains a vertex that is initially occupied by even number of searchers. This is a contradiction. Thus, when the poles are not incident $|V_o| \leq 2$.

Assume that all the vertices of G have even degree. There exists a vertex $u \in e$ such that 1 or 3 searchers start at u. This vertex has even degree only if $t(\sigma_i) = u$ for some $i \in \{1, 2, 3\}$. Thus $e' = uv'$ for some $v' \in V$. Therefore e and e' are adjacent and we have the first part.

Finally assume that there are two vertices of odd degree in G. Since an odd degree vertex is a start or an end vertex in each fast search strategy, by Lemma 8 we have that both of these vertices belong to $e \cup e'$. Let $e = uv$. If only one of u or v is odd, then e' contains the second odd vertex and the statement follows. Otherwise, either e or e' contains both of the odd vertices. Without loss of generality we can consider the case when both u and v are of odd degree. We have that the number of searchers that start at one of the vertices in e, say u, is even. Then, $\deg_G(u)$ is odd if and only if $t(\sigma_i) = u$ for some $i \in \{1, 2, 3\}$. This means that $u \in e'$. So, each pole has at least one vertex of odd degree and the lemma follows. $\square$

The following theorem completes the classification of biconnected 3-fast searchable graphs. Together with Corollary 11 they give the characterization of start and end vertices in a 3-fast search.

Theorem 10. *Let G be a reduced biconnected graph. Then G is 3-fast searchable if and only if G is outerplanar, bipolar and satisfies*

$$(V_o = \emptyset \text{ and } e \cap e' \neq \emptyset) \quad \text{or} \quad (|V_o| = 2,\ V_o \subset e \cup e', e \cap V_o \neq \emptyset \text{ and } e' \cap V_o \neq \emptyset) \tag{1}$$

for some opposing poles e and e'. Moreover, for each 3-fast search strategy for G, $\exists e, e' \in E$ such that the following holds:

$$\text{for } v \in e \setminus e', \quad \deg_G(v) \Leftrightarrow |\{\sigma_i : b(\sigma_i) = v,\ i = 1, 2, 3\}| \text{ is even, and} \tag{2}$$

$$\text{for } u \in e' \setminus e, \quad \deg_G(u) \Leftrightarrow |\{\sigma_i : t(\sigma_i) = u,\ i = 1, 2, 3\}| \text{ is even.} \tag{3}$$

Proof. If a reduced biconnected graph G is 3-fast searchable then, by Corollary 5 and by Lemma 9, G is outerplanar, bipolar and Condition (1) holds.

Let us prove the reverse implication by constructing a 3-fast search strategy for G. First, fix an outerplanar embedding of G. Since G is a reduced biconnected outerplanar bipolar graph, [14] tell us that G is 3-searchable. Then by Lemma 8, there exist two opposing poles $e = u_1 v_1$ and $e' = u_{n_1} v_{n_2}$ such that $b(\sigma_i) \in e$ and $t(\sigma_i) \in e'$ for each $i = 1, 2, 3$. Let P_1 and P_2 be the two disjoint boundary paths connecting e and e'. Let P_1 and P_2 contain the vertices $u_1, \ldots, u_{n_1}$ and $v_1, \ldots, v_{n_2}$, respectively. Assume without loss of generality that $u_1 \notin \{u_{n_1}, v_{n_2}\}$, i.e. if the poles share a vertex then this vertex is v_1.

All possible configurations of the poles (and initial and final placements of searchers) are summarized in Fig. 4. The vectors $(m; n)$ and $(k; l)$ at the vertices u_1 and v_1, respectively, characterize the possible initial configurations: m searchers start at u_1 while k searchers start at v_1 or n searchers start at u_1 while l searchers start at v_1. Note that it must hold that $m + k = n + l = 3$.

We will restrict our discussion to Case (a), the other cases follow in a similar fashion. Note that the parity of the vertices in edge e in Fig. 4(d) is independent of the fact which vertex in the pole $u_{n_1} v_{n_2}$ is of odd degree.

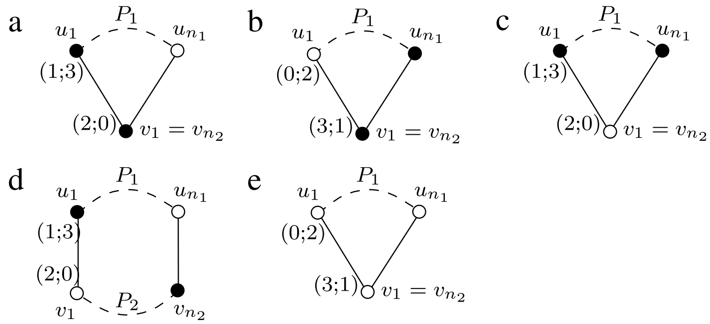

Fig. 4. All possible configurations of the poles (filled vertices are of odd degree).

Now we prove that there exists a 3-fast search strategy for G. We use induction on $i+j$ to show that if all the edges of the graph

$$G' = G[\{u_1, \ldots, u_i, v_1, \ldots, v_j\}] \tag{4}$$

have been cleared, then we can extend the cleared graph by adding one of the vertices u_{i+1}, v_{j+1} (and its corresponding edges) to the clear part of G. First we show that $G[\{u_1, v_1\}]$ can be cleared. We first clear the edge e. If $b(\sigma_1) = u_1$ and $b(\sigma_2) = b(\sigma_3) = v_1$, then move σ_3 along e to u_1, clearing e. If $b(\sigma_1) = b(\sigma_2) = b(\sigma_3) = u_1$, move σ_2 along e to v_1, clearing e. In either case, σ_3 next clears all the edges between u_1 and v_1 parallel to e. In this way $G[\{u_1, v_1\}]$ has been cleared.

Moreover, σ_1 occupies u_1 while σ_2 occupies v_1, and those searchers will follow the paths P_1 and P_2, respectively, during the search.

Assume that G' in Eq. (4) is clear. Since G is outerplanar and bipolar, we have the possibilities:

(1) All the contaminated edges incident to u_i are incident to u_{i+1}. Assume first that σ_3 is at u_i. We have $\mu(u_i, u_{i+1}) \le 2$, because otherwise the graph has a pair of nested chords. First we show that there are exactly two edges between u_i and u_{i+1}.
If $i = 1$ then it follows from how we clear $G[\{u_1, v_1\}]$ that there are exactly two searchers at u_1 if and only if $\mu(u_1, v_1)$ is odd. Since we consider the case where u_1 has odd degree and since the only neighbors of u_1 are v_1 and u_2, we must have that $\mu(u_i, u_{i+1})$ is even. Since nested chords are not allowed, $\mu(u_i, u_{i+1}) = 2$.
If $i > 1$ then, by assumption, $\deg_G(u_i)$ is even. Moreover, $\deg_{G'}(u_i)$ is also even, because each time a searcher visited and left the vertex u_i, two edges incident to u_i were cleared, and two searchers occupy u_i, which means that each of them cleared one edge incident to u_i when reaching u_i last time. So, an even number of cleared edges is incident to u_i, and consequently, an even number of contaminated edges is incident to u_i. This gives that there are exactly two contaminated edges and, by assumption, those edges are between u_i and u_{i+1}. Thus, both of the searchers at u_i can proceed to u_{i+1}. Then, σ_1 remains at u_{i+1} and σ_3 clears all edges, if any, between u_{i+1} and v_j.
We can prove similarly that if σ_3 is at v_j then $\mu(u_i, u_{i+1}) = 1$, and σ_2 can slide to u_{i+1}. Then σ_3 clears all edges between v_j and u_{i+1}.

(2) There exist contaminated edges connecting u_i and $v_{j'}$ for some $j' > j$. Since G is outerplanar and bipolar no edge $v_j u_{i'}$, $i' > i$, is possible and the situation is analogous to (1).

At this point, we have shown that a 3-fast search strategy exists for all graphs corresponding to (a) in Fig. 4. Now we show that statements (2) and (3) are true, again for case (a). Assume, by way of contradiction, that an even number of searchers begin at u_1. That is, either two or zero searchers begin there. Then, every time a searcher enters and leaves u_1, he clears an even number of edges. Since the $\deg_G(u_1)$ is odd, to clear all incident edges, at least one searcher must finish the search at u_1. However, by Lemma 8, no searcher may end at u_1, a contradiction. Again, similar arguments suffice for the other cases of Fig. 4. □

Since the search strategies described in Theorem 10 differ only by their first move, we have the following result.

Corollary 11. *If G is a reduced biconnected 3-fast searchable graph with a pair of starting and ending poles that are not adjacent, then the initial placement of the searchers is independent of the final configuration of the searchers.* □

Since any 3-fast searchable graph is 3-searchable, a complete characterization follows in the same manner as in Theorem 6 of [14], though in this case, the statement of such a theorem becomes even less pleasant. In Theorem 10 we have characterized the 3-fast searchable biconnected components of any 3-fast searchable graph.

4. NP-completeness of fast searching

In this section, we show that the fast searching problem is NP-complete. This result has been independently obtained by Yang in [17] by a direct reduction from node searching.

We introduce a new model called *weak searching* and prove that it is NP-complete, by reducing the node searching problem to weak searching. Then, we give a reduction from the weak searching problem to the fast searching problem. We consider only monotonic search strategies, so the following definitions do not allow any recontamination.

Definition 2. Given a graph G, a search strategy S is called a *monotonic k-node search strategy* if it satisfies the following:

- In the initial state all the edges of G are contaminated while in the final state all the edges of G are clear.
- Both in initial and final states no vertex of G is occupied by a searcher.
- Each step of S is one of the following.
  (i) (*placing a searcher*) Place a searcher at an arbitrary vertex of G. No more than k searchers are occupying the vertices of G at each step.
  (ii) (*removing a searcher*) Remove a searcher from a vertex v of G. This operation is allowed only if all the edges incident to v are clear or if there is another searcher located at v.
- An edge of G becomes clear whenever both of its end vertices are occupied by a searcher.

Notice that searching a multigraph does not differ from searching an underlying simple graph where parallel edges are replaced with a single edge.

Definition 3. Given a graph G a *monotonic k-weak search strategy S* is defined as follows: The initial and final states are as in a node search strategy. Each step of S consists of one of the following:

(i) (*placing a searcher*) Place a searcher at an arbitrary vertex of G. No more than k searchers are occupying the vertices of G at the end of this step.
(ii) (*removing a searcher*) Remove a searcher from a vertex v of G. This operation is allowed only if all the edges incident to v are clear or if there is another searcher located at v.
(iii) (*clearing an edge*) If at least two searchers are at u and at least one searcher is at v at the beginning of this step then we clear an edge uv by sliding one of the searchers occupying u along the edge uv. The searcher moved along uv stays at v at the end of the step.

Both models of searching we introduced above are monotonic, so we may omit the term "monotonic" when we mention these models. Note that in both the node and weak search strategy we consider the graph as searched once all the edges have been cleared and all searchers have been removed from the graph.

Using the equivalence of the pathwidth and node searching problems [12] we have the following.

Theorem 12 (*[10]*)**.** *Given an integer k and a graph G, the problem of deciding whether a monotonic k-node search for G exists is* NP-*complete.*

Theorem 13. *Given a graph G and an integer k, there exists a monotonic k-node search for G if and only if there exists a monotonic $(k+1)$-weak search for G.*

Proof. Let S be a k-node search for G using searchers $\sigma_1, \ldots, \sigma_k$. Define a $(k+1)$-weak search S' for G as follows. Initial (empty) states of S and S' are identical. If in the jth step of S a searcher σ_i, $i \in \{1, \ldots, k\}$, is placed on (removed from) $v \in V(G)$, then we extend S' by placing σ_i on v (removing σ_i from v, respectively). Moreover, if a set $X = \{vv_1, vv_2, \ldots, vv_l\}$ of edges gets clear in S as a result of placing a searcher at v, then each v_i, $i = 1, 2, \ldots, l$, must contain a searcher. In S' we clear the edges in X by adding at most $3|X|$ moves: select an edge $vv_i \in X$, place the searcher σ_{k+1} on v, slide it from v to v_i, remove it from v_i. Then remove uv_i from X. Continue repeating this procedure until all edges in X are cleared.

Assume now that S' is a $(k+1)$-weak search for G. Observe, that if in the jth step of $S'\sigma_i$ is placed on v and this results in a situation when the $k+1$ searchers are occupying $k+1$ pairwise different vertices then in the $(j+1)$th step the only allowed operation is removing a searcher from a vertex. So, we can exchange these operations. After a finite number of such modifications we obtain a $(k+1)$-weak search S'' for G. Then S'' has the property that in each step at most k different vertices are occupied by searchers. If we remove from S'' all the operations of clearing edges and the operations of placing a searcher on a vertex occupied by another searcher, then we get the desired k-node search for G. $\square$

Now we describe a polynomial-time reduction from the weak searching problem to the fast searching problem. Let k be an integer and $G = (V, E)$ be a (simple) graph such that G and k constitute an instance of the monotonic k-weak search problem. Define $G^{(t)}$ as a multigraph with t parallel edges between each pair of vertices that are adjacent in G where $t = 2(|V(G)|+1) \cdot |E(G)|$. The input to the fast searching problem is a graph $\widetilde{G}$ constructed as follows. We take $2k$ vertex disjoint copies of $G^{(t)}$, denoted by $G_1^{(t)}, \ldots, G_{2k}^{(t)}$, and we introduce another vertex v_0. For each $v \in V(G_1^{(t)}) \cup \cdots \cup V(G_{2k}^{(t)})$ there are t parallel edges between v_0 and v. Note here that the size of $\widetilde{G}$ is polynomial in the size of G, since we assume without loss of generality that $k \le |V(G)|+1$.

Theorem 14. *Let G be a connected graph and let $k \ge 2$. There exists a k-weak search for a G containing at least one edge if and only if there exists a $(k+1)$-fast search for $\widetilde{G}$.*

Proof. Let S be a k-weak search strategy for G with searchers $\sigma_1, \sigma_2, \ldots, \sigma_k$. We make five assumptions concerning S. Note that none of the assumptions on S made below increases the number of searchers used.

1. We do not move a searcher along a clear edge $e = uv$. Such an operation can be replaced by two steps: removing the searcher from u and placing it on the adjacent vertex v.
2. We do not place a searcher on a clear vertex (a vertex with no dirty edges incident to it). Such a move is not necessary, since we consider only monotone strategies, and moving a searcher along a clear edge is forbidden. This means that when all the edges adjacent to a vertex v are cleared we place no more searchers at v and we remove (not necessarily immediately) the searchers at v in the forthcoming steps.
3. All edges are cleared by the searcher σ_k. As in the proof of Theorem 13, we know that no more than $k-1$ vertices ever contain searchers at any time. Thus, there is always one searcher "free".
4. No vertex ever contains more than 2 searchers. (Prior to such an event, surplus searchers may be removed.)
5. When σ_k clears an edge uv, the move immediate before was placing σ_k at u, and the move immediately following is to remove σ_k from v.

We will construct a fast search strategy S' for $\widetilde{G}$ using $k+1$ searchers. The initial placement of the searchers in S' is $b(\sigma_i) = v_0$ for each $i = 1, \ldots, k+1$. S' proceeds in such a way that v_0 is occupied by σ_{k+1} during the entire search strategy and the subgraphs $\widetilde{G}[\{v_0\} \cup V(G_i^{(t)})]$ are cleared one by one. So, we only show how to clear $\widetilde{G}[\{v_0\} \cup V(G_1^{(t)})]$. The strategy S' for this subgraph reflects the moves done in S as follows:

1. If in S a searcher σ_i has been placed on a vertex $v \in V(G)$, then in S' we slide σ_i from v_0 to $v \in V(G_1^{(t)})$.
2. If in S a searcher σ_i ($i \neq k$) has been removed from $v \in V(G)$, then in S' we slide σ_i from $v \in V(G_1^{(t)})$ to v_0.
3. Assume that in S a searcher σ_k slides along an edge $uv \in V(G)$ from u to v. In the search strategy S', the searcher σ_k slides from v_0 to u (and in this way clears one of the parallel edges between v_0 and u). Then, σ_k clears all the edges between u and v in $\widetilde{G}$. As a result, σ_k ends at u, because there are t edges between u and v in $G_1^{(t)}$, and t is even. Then σ_k slides to v_0. Moreover, if all the edges adjacent to u (respectively v) in G are clear then we also use σ_k to clear all the edges between u (v, resp.) and v_0 of $G_1^{(t)}$. Note that after performing all above moves the searcher σ_k occupies v_0, because $\mu(v_0, x)$ is even for each $x \in V(G_1^{(t)})$.

Due to the choice of t the construction of S' is correct. In particular, note that after clearing $G_1^{(t)}$, the searchers end at v_0, ready to clear $G_2^{(t)}$, and so on.

Assume now that S' is a $(k+1)$-fast search strategy for $\widetilde{G}$. We will construct a k-weak search strategy for G. Since there are $2k$ subgraphs $G_j^{(t)}$, initially at least k of them have all the vertices unoccupied by the searchers in the strategy S'. (Otherwise, no edge could be cleared in S'.) For any such unoccupied subgraph, we know that when a searcher first reaches a vertex of $G_j^{(t)}$ then another searcher, say σ_{k+1}, is at v_0, because no recontamination is allowed. Without loss of generality, assume that the unoccupied subgraphs are $G_1^{(t)}$ to $G_k^{(t)}$.

Let σ_{k+1} be the last searcher leaving v_0 (if this ever happens in S'). Suppose that none of $G_1^{(t)}$ through $G_k^{(t)}$ have been completely cleared at this point. Then, since there is at least one contaminated edge in each of these copies of G, there must be at least two searchers present in each of the k subgraphs, accounting for $2k > k+1$ searchers, a contradiction. Thus, one of these subgraphs must have been completely cleared while the searcher σ_{k+1} remained at v_0. In the following we assume that this distinguished subgraph is $G_1^{(t)}$.

We will construct a k-weak search S for G. Remember that, according to the definition, initially no vertices of G are occupied by the searchers in S. Since $\widetilde{G}$ is a multigraph with exactly t edges between each pair of adjacent vertices u, v we know that there exists a step in S' when a searcher slides along an edge between u and v, $u, \ v \in V(G_1^{(t)})$, and two other searchers occupy u and v. We build S based on S' as follows.

1. Assume that σ_i slides from u to v, $u, \ v \in V(G_1^{(t)})$, and after this move σ_i is the only searcher at v. This means that all the remaining edges between u and v in $G_1^{(t)}$ are still dirty. In S we add two steps in which σ_i is first removed from u and then placed at v.
2. Assume that σ_i slides from u to v, $u, \ v \in V(G_1^{(t)})$, and there is a searcher at each of u and v. If this is the first time this has occurred, then in S we slide σ_i from u to v, clearing this edge. Otherwise, we remove σ_i from u and subsequently place it on v.
3. Assume that σ_i slides from u to v, $u, \ v \in V(G_1^{(t)})$, and after this move no searcher is at u. This can only occur if all the other edges between u and v in $G_1^{(t)}$ are clear; thus one of the previous moves has cleared uv in G. We add two steps to S: remove σ_i from u; place σ_i at v.

The above rules imply that the configuration of searchers $\sigma_1, \ldots, \sigma_k$ is identical in $G_1^{(t)}$ and in G after the corresponding moves in S' and S, respectively. Moreover, after each pair of corresponding steps in the search strategies, when a set of all parallel edges in $G_1^{(t)}$ is cleared, the corresponding edge in G is cleared. □

Theorem 15. *The fast searching problem is* NP*-complete for multigraphs and for simple graphs.*

Proof. Theorems 13 and 14 imply the NP-hardness of the decision version of fast searching. Due to monotonicity, the problem is clearly in NP. In order to get the result for simple graphs one can observe that there exists a k-fast search for a multigraph if and only if there exists a k-fast search for a simple graph obtained by ''dividing'' each edge of the multigraph into a path with two edges. □

5. Conclusion and future directions

We investigate some of the classical areas of graph searching as they correspond to the fast search problem; namely, we have considered graphs with small fast search number, and the complexity of the decision problem. The major area that remains unconsidered is the fast search number's relation to any of the width family of graph parameters. It is well known from [8,11] that the search number of a graph is, essentially, its pathwidth, denoted by $pw(G)$. In fact, $pw(G) \leq s(G) \leq pw(G) + 2$. As a result, we know that $pw(G) \leq s(G) \leq s_f(G)$; however, this is a terrible lower bound for fast search number, as the ratio $\frac{s_f(G)}{s(G)}$ is shown to be arbitrarily large [7]. The same example shows that treewidth is equally bad. The ''right'' width parameter remains to be seen.

Another restriction that is commonly placed on a search strategy is that at every step the set of clear edges induces a connected subgraph; such a strategy is called *connected*. We could apply the same restriction to the fast searching problem, as was briefly discussed in [7]. Investigation of the connected fast search number $cs_f(G)$ remains wide open.

Acknowledgments

We would like to thank the referees for their valuable comments and criticism.

The first author was partially supported by the Foundation for Polish Science (FNP) and by MNiSW grant N N206 379337 (2009–2011). The second author was partially supported by the Kadir Has University Research Grant BAP (2011–2013). The third author was partially supported by NSERC.

References

[1] N. Alon, P. Pralat, R. Wormald, Cleaning regular graphs with brushes, SIAM Journal on Discrete Mathematics 23 (2008) 233–250.
[2] B. Alspach, D. Dyer, D. Hanson, B. Yang, Time constrained graph searching, Theoretical Computer Science 399 (3) (2008) 158–168.
[3] L. Barrière, P. Fraigniaud, N. Santoro, D. Thilikos, Searching is not jumping, in: Proceedings of the 29th Workshop on Graph Theoretic Concepts in Computer Science, WG'03, in: LNCS, vol. 2880, Springer-Verlag, 2003, pp. 34–45.
[4] T. Biedl, T. Chan, Y. Ganjali, M.T. Hajiaghayi, D. Wood, Balanced vertex-ordering of graphs, Disrete Applied Mathematics 148 (2005) 27–48.
[5] D. Bienstock, P. Seymour, Monotonicity in graph searching, Journal of Algorithms 12 (1991) 239–245.
[6] R.L. Breisch, An intuitive approach to speleotopology, Southwestern Cavers 6 (1967) 72–78.
[7] D. Dyer, B. Yang, Ö. Yaşar, On the fast searching problem, in: R. Fleischer, J. Xu (Eds.), AAIM 2008, in: LNCS, vol. 5034, Springer-Verlag, 2008, pp. 143–154.
[8] J.A. Ellis, I.H. Sudborough, J.S. Turner, The vertex separation and search number of a graph, Information and Computation 113 (1994) 50–79.
[9] F.V. Fomin, D.M. Thilikos, An annotated bibliography on guaranteed graph searching, Theoretical Computer Science 399 (3) (2008) 236–245.
[10] J. Gustedt, On the pathwidth of chordal graphs, Discrete Applied Mathematics 45 (1993) 233–248.
[11] N.G. Kinnersley, The vertex separation number of a graph equals its path-width, Information Processing Letters 42 (1992) 345–350.
[12] L.M. Kirousis, C.H. Papadimitriou, Searching and pebbling, Theoretical Computer Science 47 (1986) 205–218.
[13] A.S. LaPaugh, Recontamination does not help to search a graph, Journal of the ACM 40 (1993) 224–245.
[14] N. Megiddo, S.L. Hakimi, M.R. Garey, D.S. Johnson, C.H. Papadimitriou, The complexity of searching a graph, Journal of the ACM 35 (1988) 18–44.
[15] T.D. Parsons, Pursuit-evasion in a graph, in: Theory and Applications of Graphs, in: Lecture Notes in Mathematics, Springer-Verlag, 1976, pp. 426–441.
[16] C. Worman, B. Yang, Searching trees with sources and targets, in: Proceedings of the 2nd Annual International Workshop on Frontiers in Algorithmics, FAW'08, in: LNCS, vol. 5059, Springer-Verlag, 2008, pp. 174–185.
[17] B. Yang, Fast edge searching and fast searching on graphs, Theoretical Computer Science 412 (2011) 1208–1219.
[18] B. Yang, D. Dyer, B. Alspach, Sweeping graphs with large clique number, Discrete Mathematics 309 (18) (2009) 5770–5780.