Discrete Mathematics 309 (2009) 5770–5780

Contents lists available at ScienceDirect

Discrete Mathematics

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

Sweeping graphs with large clique number

Boting Yang [a], Danny Dyer [b,*], Brian Alspach [c]

[a] *Department of Computer Science, University of Regina, Canada*
[b] *Department of Mathematics and Statistics, Memorial University of Newfoundland, Canada*
[c] *Department of Mathematics and Statistics, University of Regina, Canada*

ARTICLE INFO

Article history:
Received 12 June 2006
Accepted 28 May 2008
Available online 2 July 2008

Dedicated to Pavol Hell on the occasion of his sixtieth birthday

Keywords:
Edge searching
Sweeping
Clique number

ABSTRACT

Searching a network for intruders is an interesting and often difficult problem. Sweeping (or edge searching) is one such search model, in which intruders may exist anywhere along an edge. It was conjectured that graphs exist for which the connected sweep number is strictly less than the monotonic connected sweep number. We prove that this is true, and the difference can be arbitrarily large. We also show that the clique number is a lower bound on the sweep number.

1. Introduction

Sweeping (or edge searching) was originated by Parsons in [12], though the problem was of interest to spelunkers earlier than that, [5]. Parson's original problem dealt with finding a lost spelunker in a system of caves, but the problem has much wider application. We are interested in sweeping as a problem in network security, looking for methods to clean a network of a computer virus, or methods to capture a mobile intruder using software agents. In the literature, sweeping has been linked to pebbling (and hence to computer memory usage) [9], to assuring privacy when using bugged channels [8], and to VLSI (very large-scale integrated) circuit design [6]. A brief survey of results is also available [1].

We will deal primarily with graphs in which no loops or multiple edges are allowed. The number of edges incident with a vertex v of a graph G is the *degree* of v, denoted $\deg(v)$.

In this search model, collision between a searcher and an intruder may occur on an edge. This type of search is a *sweep*. The specifics of sweeping a graph G are as follows. Initially, all edges of G are *contaminated* (or *dirty*). To sweep G it is necessary to formulate and carry out a sweep strategy. A *sweep strategy* is a sequence of actions designed so that the final action leaves all edges of G *uncontaminated* (or *cleared*). In such strategies, only the following three actions are allowed, though each may occur many times.

- Place a searcher on a vertex.
- Move a single searcher along an edge uv starting at u and ending at v.
- Remove a searcher from a vertex.

A sweep strategy that restricts itself to the first two actions will be called an *internal sweep strategy*. That is, a strategy in which once the searchers are placed, they can never be removed from a vertex, but can slide along edges to other vertices.

An edge uv in G can be *cleared* in one of two ways.

* Corresponding author.
E-mail address: dyer@math.mun.ca (D. Dyer).

0012-365X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.disc.2008.05.033

- At least two searchers are placed on vertex u of edge uv, and one of them traverses the edge from u to v while the other remains at u.
- A searcher is placed on vertex u, where all edges incident with u, other than uv, are already cleared. Then the searcher moves from u to v.

A cleared edge becomes *recontaminated* if there is a path from an endpoint of the cleared edge to an endpoint of a contaminated edge, and there is no searcher anywhere on this path.

Knowing that our goal is to end up with a graph where all the edges are cleared, a basic question is: what is the least number of searchers for which a sweep strategy exists? We call this the *sweep number*, denoted $\mathrm{s}(G)$. We define the *internal sweep number* similarly and denote it $\mathrm{is}(G)$. In fact, these two numbers are equal for connected graphs. If in a sweep strategy a searcher is removed from a vertex u and placed on a vertex v, in a corresponding internal sweep the searcher may merely follow a path from u to v. We will only deal with connected graphs in this paper.

Let $E(t)$ be the set of cleared edges after the tth action in the sequence of actions that make up a sweep strategy has occurred. (Certainly, this action is one of the three actions listed above.) A sweep strategy for a graph G for which $E(t) \subseteq E(t+1)$ for all t is said to be *monotonic*. We may then define the *monotonic sweep number* and the *monotonic internal sweep number*, denoted $\mathrm{ms}(G)$ and $\mathrm{mis}(G)$, respectively. Similarly, a sweep strategy such that $E(t)$ induces a connected subgraph for all t is said to be *connected*, and we may define the *connected sweep number* $\mathrm{cs}(G)$ and the *connected internal sweep number* $\mathrm{ics}(G)$. Finally, a sweep strategy may be both connected and monotonic, giving us the *monotonic connected sweep number* $\mathrm{mcs}(G)$ and the *monotonic connected internal sweep number* $\mathrm{mics}(G)$.

LaPaugh [10] and Bienstock and Seymour [4] proved that for any connected graph G, $\mathrm{s}(G) = \mathrm{ms}(G)$. Barrière et al. [3] extended this result, giving the following relations for these numbers.

Theorem 1.1. *For any connected graph G,*

$$\mathrm{is}(G) = \mathrm{s}(G) = \mathrm{ms}(G) \leqq \mathrm{mis}(G) \leqq \mathrm{cs}(G) = \mathrm{ics}(G) \leqq \mathrm{mcs}(G) = \mathrm{mics}(G).$$

This chain of inequalities suggests several questions. For instance, can equality be achieved? Do graphs exist for which the inequalities are strict?

An example in [3] shows that the first inequality may be strict. The graph below, which we call the "Y-square", is another example. Moreover, this is an example with fewer vertices and edges than the example in [3]. For this example, the sweep number is 3, while the monotonic internal sweep number is 4. We conjecture that this is the smallest graph that exhibits the strict inequality between sweep number and monotonic internal sweep number.

The second inequality, $\mathrm{mis}(G) \leqq \mathrm{cs}(G)$, was also proved in [3]. Further, the authors gave an example showing that the inequality was strict. With this result, they also observed that, generally, the monotonic internal sweep number or connected sweep number of a graph G may be smaller than the monotonic internal sweep number or connected sweep number of some minors of G. We prove these results by using large cliques as our building blocks, thereby allowing us to calculate the sweep numbers easily.

Whether the third inequality, $\mathrm{cs}(G) \leqq \mathrm{mcs}(G)$, can be strict was left as an open problem in [3]. We will show that there exists a graph G such that $\mathrm{cs}(G) < \mathrm{mcs}(G)$, and that, in fact, the difference between these two values can be arbitrarily large.

We will also show that $\mathrm{is}(K_n) = \mathrm{mics}(K_n) = n \geqq 4$, where K_n is the complete graph on n vertices. This means that there is exactly one sweep number for complete graphs.

In general, determining the sweep number of a graph G is NP-complete [11]. As any successful sweep strategy gives an upper bound, our goal becomes first to find the "right" way to clear the graph, using as few searchers as possible. Once this strategy is found, we must then prove that no fewer searchers will suffice. Here is where the true difficulty lies: most easily attainable lower bounds are quite poor. We will prove several lower bound results using the clique number of a graph. Some of this work has appeared in preliminary form in [14].

Definition 1.2. The *clique number* of the graph G, denoted $\omega(G)$, is the largest number such that G contains a clique of order $\omega(G)$.

2. Sweeping and cliques

The main result of this section is Theorem 2.4. We use it to prove several lower bounds for the sweep number.

Definition 2.1. A vertex in a graph G is said to be *exposed* if it has edges incident with it that are contaminated as well as edges incident with it that are cleared. Following a sweep strategy S on G, we define $\mathrm{ex}_S(G, i)$ to be the number of exposed vertices after the ith step.

Definition 2.2. A vertex v is said to be *cleared* if all the edges incident with it are currently uncontaminated.

The following obvious lemma is a generalization of a result from [11].

Lemma 2.3. *At the time the first vertex v becomes cleared in a graph G, there must be a searcher on each neighbour of v.*

It is easy to see that $\mathrm{s}(K_1) = 1$, $\mathrm{s}(K_2) = 1$, and $\mathrm{s}(K_3) = 2$, and only slightly more difficult to see that $\mathrm{s}(K_4) = 4$.

The following result gives several useful corollaries. For a graph G, we will denote the minimum degree of G by $\delta(G)$.

Theorem 2.4. *If G is connected and $\delta(G) \geqq 3$, then $\mathrm{s}(G) \geqq \delta(G) + 1$.*

Proof. Consider a graph G with minimum degree $\delta(G)$, and a sweep strategy S that clears it. If the first vertex cleared by S is not of minimum degree, then it must have at least $\delta(G) + 1$ vertices adjacent to it. When it is cleared, each of the these vertices must contain a searcher and $\mathrm{s}(G) \geqq \delta(G) + 1$.

We now consider the last time that the graph goes from having no cleared vertices to a single cleared vertex u. By the preceding paragraph, we may assume that u is a vertex of minimum degree. We will assume that the strategy S employs at most $\delta = \delta(G)$ searchers, and arrive at a contradiction. Let the neighbours of u be denoted $v_1, v_2, \ldots, v_\delta$. Assume, without loss of generality, that uv_1 is the final edge incident with u cleared, and that uv_2 is the penultimate such edge.

Consider the placement of searchers the moment before uv_1 is cleared. Since each of uv_i, $2 \leqq i \leqq \delta$, is cleared, there are searchers on each end vertex of these edges and on u. But this uses all δ searchers. Thus the only way that uv_1 can be cleared is if the searcher at u traverses the edge uv_1 from u to v_1. Thus, all the other edges incident with v_1 are contaminated. Since $\delta \geqq 3$, the searcher on v_1 cannot move.

Now consider the placement of searchers before the penultimate edge uv_2 is cleared. Again, as each of the edges uv_i, $3 \leqq i \leqq \delta$, is cleared, there is a searcher on each end vertex of these edges and on u. This accounts for $\delta - 1$ searchers. Since the next move is to clear uv_2, the single free searcher must be on either u or v_2. Moving from v_2 to u would instantly recontaminate the edge uv_2 which implies the edge must be cleared from u to v_2. This leaves the searcher at v_2, and all the other edges incident with v_2 must be contaminated. Since $\delta \geqq 3$, the searcher on v_2 cannot move.

Consider a searcher on v_i, $3 \leqq i \leqq \delta$. If the vertex v_i is adjacent to v_1 and v_2, then the edges v_1v_i and v_2v_i are contaminated, and the searcher at v_i cannot move.

If the vertex v_i is adjacent to exactly one of v_1 and v_2, it must also be adjacent to some other vertex w not adjacent to u (as the degree of v_i is at least δ). As there is no searcher on w, the only way that v_iw can be cleared is if w is a cleared vertex. However, we know that u is the first cleared vertex, so that w is not cleared. Thus, the searcher at v_i cannot move.

Finally, if the vertex v_i is adjacent to neither v_1 nor v_2, it must be adjacent to two vertices w_1 and w_2 neither of which is adjacent to u. As before, these edges cannot be cleared, and thus the searcher at v_i cannot move.

As there are still contaminated edges, and none of the δ searchers can move, we have obtained the required contradiction. ■

Corollary 2.5. *For a connected graph G, let $\kappa(G)$ be the vertex connectivity and $\kappa'(G)$ be the edge connectivity of G. If $\kappa(G) \geqq 3$, then $\mathrm{s}(G) \geqq \kappa'(G) + 1 \geqq \kappa(G) + 1$.*

Corollary 2.6. *For all positive integers $n \geqq 4$, $\mathrm{s}(K_n) = \mathrm{mis}(K_n) = \mathrm{cs}(K_n) = \mathrm{mcs}(K_n) = n$.*

Proof. By Theorem 2.4, we know that $\mathrm{s}(K_n) \geqq n$. We present the following monotonic connected sweep strategy for K_n using n searchers. First, clear a vertex v of K_n. This requires $n - 1$ searchers, leaving one free. This free searcher may then clear all the edges of K_n that are not incident with v. By Theorem 1.1, we are done. ■

Definition 2.7. The *wheel graph* W_n, $n \geqq 3$, is the graph formed by connecting all the vertices of an n-cycle to another vertex v not on the cycle.

Using Theorem 2.4, it is similarly easy to prove the following.

Corollary 2.8. *For all $n \geqq 3$, $\mathrm{s}(W_n) = \mathrm{mis}(W_n) = \mathrm{cs}(W_n) = \mathrm{mcs}(W_n) = 4$.*

Theorem 2.9. *If a graph H is a minor of a graph G, then $\mathrm{s}(H) \leqq \mathrm{s}(G)$.*

Proof. Let $\Phi : V(G) \to V(H)$ denote the function that maps the vertices of G to the corresponding vertices of H that result from vertex identifications that have taken place to form the minor H. Suppose that $\mathrm{s}(G) = k$. Whenever a searcher in G moves from a vertex u along an edge to a vertex v, the corresponding searcher does nothing in H when $\Phi(u) = \Phi(v)$. If $\Phi(u) \neq \Phi(v)$, then the corresponding searcher does nothing when $\Phi(u)$ and $\Phi(v)$ are not adjacent in Y, but traverses the edge from $\Phi(u)$ to $\Phi(v)$ when they are adjacent in Y. It is easy to see that k searchers clear all of Y if they clear X. The result follows. ■

The following lemma is straightforward, and gives a trivial upper bound for the sweep number.

Lemma 2.10. *If G is a connected graph, then $\mathrm{s}(G) \leqq \min(|V(G)|, |E(G)|)$.*

If we consider the graph K_3, which has sweep number 2, then the graph obtained from K_4 by removing a single edge has sweep number 3, and is the unique supergraph of K_3 with the least number of edges such that its sweep number is 3. (The same cannot be said for K_2, which is contained in the supergraphs K_3 and the star $K_{1,3}$, both of which have sweep number 2.) For larger n, we have the following theorem.

Theorem 2.11. *For $n \geqq 4$, K_{n+1} is the unique connected supergraph of K_n with the least number of edges such that its sweep number is $n+1$.*

Proof. First, note that K_{n+1} is a supergraph of K_n with sweep number $n+1$, and K_{n+1} has n additional edges. We will show that any other supergraph G containing clique of order n with at most n additional edges satisfies $\mathrm{s}(G) = n$. Denote the clique of order n in G by H.

If G has only $k < n$ additional edges than those in H, consider the connected components of the graph induced by $E(G) - E(H)$. Place one searcher on each of the k edges of these components, and sweep these components as in Lemma 2.10, ending on vertices of H. Place a searcher on one of these vertices, and clear all the edges induced by them. Place the remaining $n-k-1$ searchers on an arbitrary one of these vertices, and use these searchers to clear it. This leaves a searcher on every other vertex of H, and one free searcher. This searcher may clear all remaining edges. Thus, $\mathrm{s}(G) = n$.

If G has n additional edges than those in H, and some vertex $v \in H$ is not incident with any of these edges, then as before, clear the additional edges as in Lemma 2.10, ending with searchers on k vertices of H. There are at most $n-1$ such exposed vertices (since v is adjacent to no additional edge) and so there is a free searcher to clear all edges between these vertices. Then, place $n-k-1$ searchers on an arbitrary one of the k vertices, and use these searchers to clear this vertex. This leaves a searcher on the $n-1$ other vertices of H, and the remaining searcher can clear the graph. Thus, $\mathrm{s}(G) = n$.

Finally, we consider when G has n additional edges than those in H, and every vertex of H is incident with exactly one of these edges. If G has more than one vertex not in $V(H)$, let u be one such vertex. To sweep G, first clear u. Use a free searcher to clear all the edges induced by the k neighbours of u, then place $n-k-1$ searchers on an arbitrarily chosen one of these k vertices. Clear that vertex. This leaves a searcher on every other vertex of H, and one free searcher. Use this searcher to clear the edges of H. Then the neighbours of every vertex in $V(G) - V(H)$ contain a searcher, and these searchers may clear these vertices, clearing G. Thus, $\mathrm{s}(G) = n$. ∎

Theorem 2.12. *If $n \geqq 4$, then the graph of order n with the greatest number of edges and sweep number $n-1$ is the complete graph K_n with one edge removed.*

Proof. Let $K_n - \{uv\}$ denote the complete graph of order n with the edge uv deleted. We first use $n-2$ searchers to clear the vertex v and station these searchers on the $n-2$ neighbours of v. Then we use one free searcher to clear all the contaminated edges between the $n-2$ neighbours of v. Finally, we use $n-1$ searchers to clear all the remaining contaminated edges incident on u. Thus, $K_n - \{uv\}$ is $(n-1)$-sweepable. On the other hand, from Theorem 2.4, we have $\mathrm{s}(K_n - \{uv\}) \geqq n-1$. Therefore, $\mathrm{s}(K_n - \{uv\}) = n-1$. ∎

Definition 2.13. The *cartesian product* of two graphs G and H, denoted $G\Box H$, is a graph with vertex set $V(G) \times V(H)$. Two vertices (u_1, v_1) and (u_2, v_2) are adjacent in $G\Box H$ if and only if $u_1 = u_2$ and v_1v_2 or $u_1u_2 \in E(H)$ and $v_1 = v_2$.

The sweep number of the cartesian product of graphs has also been considered in [13], where the following result is proved.

Theorem 2.14. *For two connected graphs G and H,*

$$\mathrm{s}(G\Box H) \leqq \min(|V(G)| \cdot \mathrm{s}(H), |V(H)| \cdot \mathrm{s}(G)) + 1.$$

Corollary 2.15. *If G is a connected graph and $n \geqq 4$, then*

$$\mathrm{s}(K_n\Box G) \leqq n \cdot \mathrm{s}(G) + 1.$$

In fact, it is easy to see that we can do better than this when G is also a complete graph.

Corollary 2.16. *For $n \geqq 1$ and $m \geqq 2$, $\mathrm{mcs}(K_n\Box K_m) \leqq n(m-1) + 1$.*

Proof. Let $u_1, u_2, \ldots, u_n$ be the vertices of K_n, and $v_1, v_2, \ldots, v_m$ be the vertices of K_m. Place one searcher on each of the (u_i, v_1), and the remaining searchers anywhere on any of these same vertices. Use a free searcher to clear all the edges in the clique induced by $\{(u_i, v_1)\}$. There is a perfect matching between the clique induced by $\{(u_i, v_1)\}$ and the clique induced by $\{(u_i, v_2)\}$. Move n searchers to clear the perfect matching by moving one searcher along each of the edges $((u_i, v_1), (u_i, v_2))$. Similarly, searchers can traverse perfect matchings from the clique induced by $\{(u_i, v_1)\}$ to the clique induced by $\{(u_i, v_j)\}$, $3 \leqq j \leqq m$. This leaves $n(m-1)$ searchers stationed on $\{(u_i, v_j)\}$, $2 \leqq j \leqq m$, and the remaining free searcher can clear all the edges between these vertices. ∎

In the special case that exactly one of the complete graphs is K_2, we can say something even more precise.

Corollary 2.17. *If $n \geqq 3$, then $\mathrm{s}(K_n\Box K_2) = n + 1$.*

3. Differences between sweep numbers

From Corollary 2.6 and Theorem 2.9, we obtain the following result.

Theorem 3.1. *For any graph G, if* $\omega(G) \geqq 4$*, then* $\omega(G) \leqq \mathrm{s}(G)$.

Since trees have clique number 2 and there exist trees with arbitrarily large sweep numbers [2], it might appear that the bound presented in Theorem 3.1 is not particularly useful. This is not the case, as Theorem 3.1 provides a basis for constructing graphs with easily calculated sweep numbers.

We will prove that the difference between the connected sweep number and the monotonic connected sweep number of a graph may be arbitrarily large. To this end, we first construct a graph W which will demonstrate that these two sweep numbers may be different, and we then use W to construct an infinite family of graphs in which the difference between these two sweep numbers becomes arbitrarily large. Similarly, using the Y-square and constructed graphs X and Y, we construct infinite families of graphs in which the connected sweep number is arbitrarily larger than the monotonic internal sweep number, and the monotonic internal sweep number is arbitrarily larger than the sweep number.

We construct the graph W as shown in Fig. 2. In this figure, a circle represents a complete graph on the indicated number of vertices, and double lines between two cliques A and B indicate a perfect matching either between A and B (if $|A| = |B|$) or between A and a subgraph of B (if $|A| < |B|$). The latter is called a *saturated matching*.

If there is a saturated matching from a graph A to a subgraph of B, we use $B[A]$ to denote the graph induced by those vertices of B adjacent to vertices of A. So $B[A]$ is also a clique.

We construct W such that $A_9[C_{19}]$, $A_9[D_{19}]$, $A_9[E_{300}]$ and $A_9[F_{300}]$ may be chosen to be vertex-disjoint, and similarly for A'_9. Also, $V(A_2[C_1]) \cap V(A_2[B_1]) = \emptyset$ and $V(A_4[D_1]) \cap V(A_4[B_{300}]) = \emptyset$, and similarly for A'_2 and A'_4. Finally, there are 300 cliques between A_1 and A'_1, each of which contains 280 vertices.

Theorem 3.2. *For the graph* W*,* $\mathrm{cs}(W) = 281$*.*

Proof. It follows from Corollary 2.6 that $\mathrm{cs}(A_9) = \mathrm{cs}(A'_9) = 281$ and from Theorem 3.1 that we need at least 281 searchers to clear W. To prove that this number is sufficient, we sketch a sweep strategy using the same number of searchers.

We begin by clearing a vertex in A'_9. First, place all 281 searchers on a single vertex v of $A'_9 \setminus (A'_9[C'_{19}] \cup A'_9[D'_{19}] \cup A'_9[E'_{300}] \cup A'_9[F'_{300}])$. Move 280 of them to the 280 neighbours of v. This clears v, and the single searcher remaining on v then clears all remaining edges in A'_9.

Then clear the cliques of C', D', E', and F', ending with searchers stationed on appropriate vertices of A'_2, A'_4, A'_5, and A'_6. Using the free searchers, clear A'_8, and then A'_6. Stationing searchers on all the vertices of A'_6 to prevent recontamination, we use the remaining searchers to clear A'_7 and then A'_5, stationing searchers on A'_5 to prevent recontamination. This leaves a sufficient number of searchers to clean the T'_i.

With the free searchers remaining and those stationed on A'_6, clear the L'_i, then A'_2, stationing searchers there after wards. This leaves sufficient searchers to clear the R'_i, then A'_4, stationing searchers at A'_4. This leaves sufficient searchers to clear the B'_i, which, once cleared, allow for enough free searchers to clear A'_3.

We now use 281 searchers to clear, one by one, the 300 cliques between A'_1 and A_1, followed by A_1 itself. Then we move the searchers from $A_1[A_2]$ to A_2, and use a free searcher to clear all edges in A_2. We now have 80 searchers stationed in A_2, leaving 201 free searchers.

Pick a vertex in C_1 and move a searcher to this vertex from $A_2[C_1]$. Then move another searcher along this edge, and to the corresponding vertex in C_2. Then another to the corresponding vertex in C_3, and so on, until finally we have placed a searcher on the corresponding vertex in $A_9[C_{19}]$. Then move a searcher to a vertex in $A_9[D_{19}]$, followed by moving a searcher to a corresponding vertex in D_{19}, then another to a corresponding vertex in D_{18}, and so on, until reaching the corresponding vertex in D_1. Finally, move one searcher to the corresponding vertex in $A_4[D_1]$. We now have 80 searchers stationed in A_2, and in total, 41 searchers along a path through the C_i, through A_9, and finally through the D_i into A_4. This leaves 160 free searchers.

Move these free searchers along this path, to the single vertex in A_3 adjacent to the path. Clear this vertex, and then use the single free searcher to clear A_3. Then the searchers on $A_3[A_4]$ move to A_4, and a free searcher can clear A_4. With 110 searchers stationed on A_4, 80 searchers stationed on A_2, and 40 searchers strung in that path from A_1 to A_4 through A_9, there are 51 free searchers. These searchers can clear the B_i.

We now collapse the path from A_2 to A_4 through A_9, in the following manner. First, we remove the searcher in C_1; then remove the searcher in C_2, and so on, until finally we remove the searcher in D_1. These searchers may then be placed on any vertex in A_4.

Using the searchers not stationed at A_2 and A_4, clear the D_i, stationing searchers at $A_9[D_{19}]$. Use the remaining free searchers to clear the R_i, ending at A_5. Leaving searchers stationed at A_5, we clear the C_i, ending at $A_9[C_{19}]$, and then clear the L_i, eventually ending with searchers stationed at A_6. This leaves enough free searchers to clear the T_i, then the F_i, which in turn leaves enough free searchers to clear A_7. These free searchers can be used to clear the E_i, then A_8, and finally A_9. ∎

It is important to note that the strategy demonstrated in Theorem 3.2 is not monotonic, as edges in the path from A_2 through A_9 to A_4 were allowed to be recontaminated.

Theorem 3.3. *For the graph W,* $\mathrm{mcs}(W) = 290$.

Proof. It is straightforward to show that 290 searchers are sufficient to clear W in a monotonic connected fashion; essentially, the same sweep strategy as Theorem 3.2 may be used, with a single change. Instead of clearing a path from A_2 through A_9 to A_4 to clear A_3 and A_4, now there are sufficient searchers to clear the B_i instead. After A_3 and A_4 are cleared, the original strategy suffices.

To prove the equality, we will show that $\mathrm{mcs}(W) > 289$. First, assume that W is 289-monotonically connected sweepable. Let S be a monotonic connected sweep strategy using 289 searchers. Because of the involution automorphism interchanging the right side and left side, we may assume the first cleared edge lies to the left of the 280-clique G_{150} or has one end vertex in G_{150}.

We will make heavy use of vertex-disjoint paths determined by perfect matchings between successive cliques. The most important family of such paths is $P_1, P_2, \ldots, P_{80}$ consisting of the 80 vertex-disjoint paths having one end vertex in A_2 and the other end vertex in G_{150} along the chain of connecting 280-cliques.

We call a clique *pseudo-cleared* if it contains exactly one cleared vertex. We are interested in which of A_3, A_8 or A_9 is the first to be pseudo-cleared. Suppose A_9 is the first of the three to be pseudo-cleared. At the moment the first vertex of A_9 is cleared, there must be 280 exposed vertices in A_9. Since this sweep is connected, there must be a path Q from G_{150} to A_9 in the subgraph of cleared edges. The path Q must pass through at least nineteen 20-cliques. Since there are at most 9 additional exposed vertices, at least one of the 20-cliques, call it K, through which Q passes is cleared. From K, there are 20 vertex-disjoint paths back to A_2 not passing through A_9. Without loss of generality, assume these 20 vertex-disjoint paths terminate at the end vertices of $P_1, P_2, \ldots, P_{20}$. Call the extensions of $P_1, P_2, \ldots, P_{20}$ to K by $Q_1, Q_2, \ldots, Q_{20}$.

For each Q_i, $1 \leqq i \leqq 20$, we examine what happens as we start working back from K along the path Q_i. Since the clique K is cleared, the last vertex of Q_i (the one in K) is cleared. That is, the last edge of Q_i is cleared. Move to the preceding vertex. If it is not cleared, then we have encountered an exposed vertex on Q_i. If it is cleared, then we move to the preceding vertex on Q_i.

If Q_i passes through either A_6 or A_4, either we encounter an exposed vertex or the vertex u_i of Q_i in A_6 or A_4 is cleared. But if the latter is the case, then the edge from u_i to A_8 or A_3 is cleared. Since neither A_8 nor A_3 have any cleared vertices, we have found an exposed vertex corresponding to the path Q_i.

If Q_i does not pass through A_4 or A_6, then either we encounter an exposed vertex or we reach the vertex u_i of Q_i in A_2, with u_i cleared. But now we extend a path from u_i through the L_j-cliques to A_8 and we must eventually encounter an exposed vertex.

Therefore, each path Q_i yields a distinct exposed vertex giving us at least 300 exposed vertices. We now see that A_9 cannot be the first pseudo-cleared clique amongst A_3, A_8, and A_9. Similar arguments concerning the number of vertex-disjoint paths between cleared cliques and contaminated cliques show that neither of A_3 or A_8 can be cleared first, thus reaching a contradiction. ■

Theorems 3.2 and 3.3 give the following result.

Corollary 3.4. *There exists a graph G such that* $\mathrm{cs}(G) < \mathrm{mcs}(G)$.

We have shown that the difference between $\mathrm{cs}(W)$ and $\mathrm{mcs}(W)$ was 9, though in fact, the difference can be smaller. For instance, we could reduce the size of cliques and length of "paths" by an approximate factor of 5 and then prove that $\mathrm{cs}(W_{\frac{1}{5}}) = 57$ and $\mathrm{mcs}(W_{\frac{1}{5}}) = 58$ (This corresponds to letting $k = \frac{1}{5}$ in the following construction.) However, Corollary 3.4, while valid, is more easily demonstrated by using larger cliques and longer paths to make the difference more believable.

From the graph W, we may define a family W_k of graphs constructed as follows for $k \geqq 1$. In W_k, the 300 cliques of the R_i (and R'_i) are replaced by $300k$ cliques of order $180k$. Similarly, the T_i (and T'_i) are replaced by $300k$ cliques of order $80k$, the B_i (and B'_i) are replaced by $300k$ cliques of order $50k$, the L_i (and L'_i) are replaced by $300k$ cliques of order $160k$, the E_i (and E'_i) are replaced by $300k$ cliques of order $20k$, the F_i (and F'_i) are replaced by $300k$ cliques of order $20k$, and the G_i are replaced by $300k$ cliques of order $280k$. The cliques $A_2, A_5, A_6, A'_2, A'_5$, and A'_6 are replaced by cliques of order $80k$. The cliques A_4 and A'_4 are replaced by cliques of order $110k$. The cliques A_3 and A'_3 are replaced by cliques of order $160k$. The cliques A_1 and A'_1 are replaced by cliques of order $280k$. The 19 cliques that make up the C_i are replaced by $20k - 1$ cliques of order $20k$. Similarly for the C'_i, D_i, and D'_i. The cliques A_8 and A'_8 are replaced by cliques of order $200k + 1$, the cliques A_7 and A'_7 are replaced by cliques of order $140k + 1$, and the cliques A_9 and A'_9 are replaced by cliques of order $280k + 1$.

The resulting graphs W_k are "scaled up" version of W, with appropriately changed sweep numbers. Using an argument similar to those in the proofs of Theorems 3.2 and 3.3, we can prove the following theorem.

Theorem 3.5. *For $k \geqq 1$,* $\mathrm{cs}(W_k) = 280k + 1$, *and* $\mathrm{mcs}(W_k) = 290k$.

We have exhibited the utility of cliques in constructing the graph W. We will now extend this technique to other graphs so that we may study other properties of sweeping.

Referring to Fig. 3, let $X' = K_{10} \square P_{60}$. In the graph X, by construction, $V(X_{21}[A_{20}]) \cap V(X_{21}[B_{20}]) = \emptyset = V(X_{40}[C_{41}]) \cap V(X_{40}[D_{41}])$. It is easy to see that X is a subgraph of X'.

Theorem 3.6. *For X as pictured in Fig. 3,*

$$s(X) < \mathrm{mis}(X) < \mathrm{cs}(X).$$

Proof. Recall from Corollary 2.17 that $s(K_{10}\Box K_2) = 11$. Since X contains $K_{10}\Box K_2$ as a minor, by Theorem 2.9 we know that $s(X) \geqq 11$. In fact, we can use 11 searchers to clear X, by first placing 5 searchers on A_1 and 5 searchers on B_1. Then a single free searcher can be used to clear all the edges in A_1. Then 5 searchers move along the perfect matching to A_2, and a single free searcher clears all the edges in A_2, and so on, finally reaching $X_{21}[A_{20}]$. The single free searcher moves to B_1, and the process is repeated, clearing the B_i and moving to $X_{21}[B_{20}]$. With 10 searchers on X_{21}, a single free searcher may then clear all the edges of X_{21}. These 11 searchers may then clear the X_i clique by clique, finally reaching X_{40}. Stationing 5 searchers on $X_{40}[D_{41}]$, the remaining 6 searchers may clear the C_i. Then the D_i may be cleared. Thus, $s(X) = 11$.

The graph X can be cleared by 16 searchers in a connected sweep. Placing all 16 searchers on A_1, we may use one free searcher to clear the edges of A_1. Then 5 searchers may move to A_2, and a free searcher may clear the edges of A_2, and so on, until finally $X_{21}[A_{20}]$ is cleared. Then we may place 5 more searchers on the remaining vertices of X_{21}, and use a free searcher to clear the remaining edges in X_{21}. Leaving a searcher on each vertex of X_{21}, there are 6 free searchers. These searchers may clear the B_i clique by clique. Then the 10 searchers on X_{21} plus another free searcher may clear the X_i through X_{40}. Finally station 10 searchers on X_{40}. This leaves 6 free searchers, who can be used to clear the C_i and the D_i. Thus, $\mathrm{cs}(X) \leqq 16$. Considering vertex-disjoint paths as in Theorem 3.3, it can be shown that 15 searchers are insufficient to clear X in a connected sweep.

To obtain a monotonic internal sweep, first place 6 searchers on A_1, and 6 searchers on B_1. The 6 searchers on A_1 can clear the A_i, eventually stationing 5 searchers on $X_{21}[A_{20}]$. The 6 searchers on B_1 can clear the B_i, eventually stationing 5 searchers on $X_{21}[B_{20}]$. Then we may follow the same strategy as the sweep above. Thus $\mathrm{mis}(X) \leqq 12$. The proof that 12 searchers are necessary follows in a similar fashion. ■

We now consider the graph X'.

Lemma 3.7. *For the graphs X and X',* $\mathrm{cs}(X) > \mathrm{cs}(X')$.

Proof. Since X' contains $K_{10}\Box K_2$ as a minor, we know by Theorem 2.9 and Corollary 2.17 that $\mathrm{cs}(X') \geqq 11$. In fact, we can use 11 searchers in a connected sweep to clear X', by placing 10 searchers on X_1, and then using a single free searcher to clear all the edges in X_1. Then 10 searchers move along the perfect matching to X_2, and a single free searcher clears all the edges in X_2, and so on, finally reaching X_{60}. Thus, $\mathrm{cs}(X') = 11$.

From the proof of Theorem 3.6, we know that $\mathrm{cs}(X) = 16$, and the result follows. ■

Since X is a subgraph of X', this lemma has an immediate consequence, as observed in [3]. If H is a minor of a graph G, then in contrast to Theorem 2.9, it does not follow that $\mathrm{cs}(H) \leqq \mathrm{cs}(G)$.

Continuing in the same vein, let $Y' = K_{10}\Box P_{120}$, and Y be as pictured in Fig. 4, where circles and double lines are defined as above. It is easy to see that Y is a subgraph of Y'.

Theorem 3.8. *For graphs Y and Y' as given,* $\mathrm{mis}(Y) > \mathrm{mis}(Y')$.

Proof. We first note that $K_{10}\Box K_2$ is a subgraph of Y', and thus $11 \leqq \mathrm{mis}(Y')$. Also, Y' can be cleared using the same strategy as used for X' in Lemma 3.7. Thus, $\mathrm{mis}(Y') = 11$.

The graph Y can be cleared by 16 searchers in a monotonic internal fashion. Place 16 searchers on A_1. Use 6 searchers to clear the A_i, stationing 10 searchers on Y_{21}. Use the six remaining searchers to clear the B_i. Clear to Y_{40}, stationing 10 searchers on Y_{40}. This leaves 6 free searchers that can be used to clear the E_i. Then the searchers may clear to Y_{80}, stationing 10 searchers there. The remaining 6 free searchers may clear the F_i. Then all the searchers may clear to Y_{100}, stationing 10 searchers there. The 6 remaining searchers may clear the C_i and then the D_i. Thus, $\mathrm{mis}(Y) \leqq 16$. Using vertex-disjoint paths as in Theorem 3.3, it can be shown that 15 searchers are insufficient to clear X. ■

As before, since Y is a subgraph of Y', there is an immediate corollary, as observed in [3]. In contrast to Theorem 2.9, if H is a minor of G, then it does not follow that $\mathrm{mis}(H) \leqq \mathrm{mis}(G)$.

As with W, we may create families of graphs X_k (and X'_k) and Y_k (and Y'_k) based on X (and X') and Y (and Y'). This is done by replacing cliques of order 5 with cliques of order $5k$ and cliques of order 10 with cliques of order $10k$, and lengthening "paths" of cliques of the same order by a factor of k. (For instance, in X, rather than having 20 cliques of order 10 make up the X_i, they would be replaced by $20k$ cliques of order $10k$.) The results for these families are summarized below.

Theorem 3.9. *For $k \geqq 1$,* $\mathrm{cs}(X_k) = 15k+1$*;* $\mathrm{mis}(X_k) = 10k+2$*;* $\mathrm{cs}(X'_k) = 10k+1$*;* $\mathrm{mis}(Y) = 15k+1$*; and* $\mathrm{mis}(Y') = 10k+1$.

This result tells us that the difference between the monotonic internal sweep number of a graph and the connected sweep number can be large. As well, the results tell us that in the case of monotonic internal and connected sweeps, a subgraph may need arbitrarily more searchers than the supergraph.

The graph in Fig. 5 is a similarly "scaled up" version of the Y-square (pictured in Fig. 1). Here, edges are replaced by "paths" of cliques, with each path containing k^2 cliques of size k. This increases the sweep number to $3k + 1$, and the monotonic internal sweep number to $4k$, again showing that the difference in these values can be quite large.

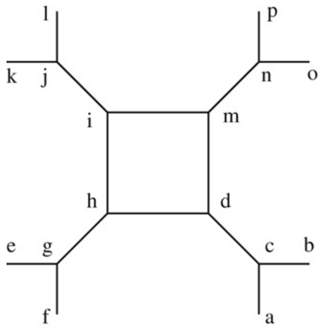

Fig. 1. The Y-square.

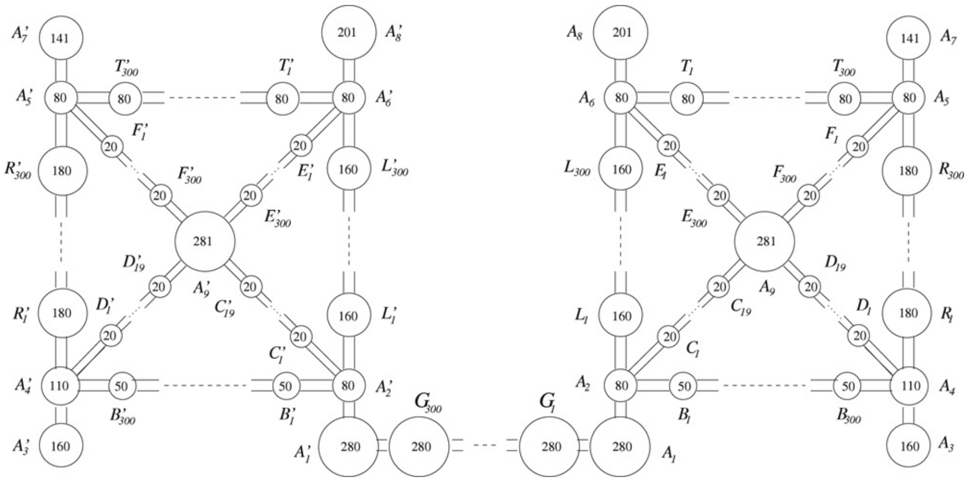

Fig. 2. The graph W.

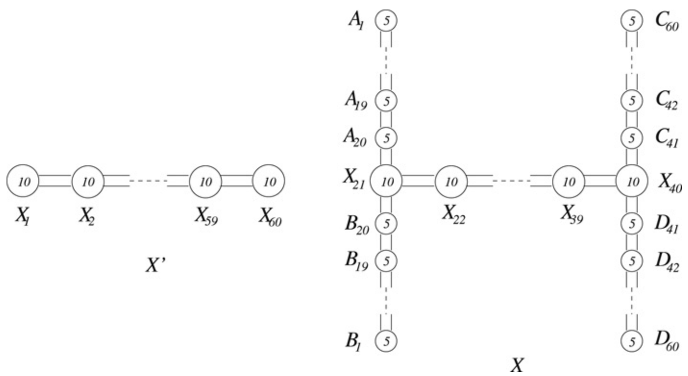

Fig. 3. The graph X' and its subgraph X.

Recall that there are three inequalities in Theorem 1.1. Corollary 3.4 shows that the final inequality can be strict, and Theorem 3.6 shows that the first pair may also be strict. This leads us to construct a single graph H for which the three inequalities strictly hold (see Fig. 6).

Theorem 3.10. *For the graph H as given,* $\mathrm{s}(H) < \mathrm{mis}(H) < \mathrm{cs}(H) < \mathrm{mcs}(H)$.

This graph H has $\mathrm{s}(H) = 561$, $\mathrm{mis}(H) = 570$, $\mathrm{cs}(H) = 841$, and $\mathrm{mcs}(H) = 850$. The proofs of these claims follow in the same manner as the proofs of Theorems 3.2, 3.3 and 3.6.

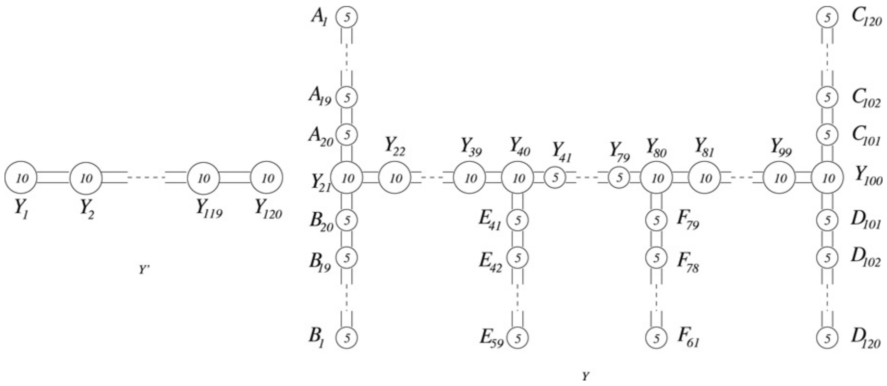

Fig. 4. The graph Y' and its subgraph Y.

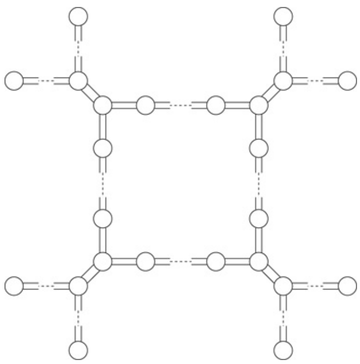

Fig. 5. The kY-square.

4. Variation on the required number of searchers

We know the maximum number of exposed vertices in a connected graph is at most the sweep number of the graph. Of course, most of the time, the number of exposed vertices is much less than this maximum. In a real world situation, most searchers not on exposed vertices could "go away", and would only return when needed. So we would be interested in sweep strategies that minimizes the number of exposed vertices at each step. For a graph G, the sequence of $\mathrm{ex}_S(G, i)$ for any S could vary greatly. The following theorem illustrates just how great this variance can be.

We first construct a graph Z as pictured in Fig. 7, where the a_i are positive integers, and $M = \max_{1 \leqq i \leqq n} a_i + 5$. (The value 5 is added for safety.)

Theorem 4.1. *Given a finite sequence of positive integers, $a_1, a_2, \ldots, a_n$, then with Z as given, every monotonic connected sweep strategy S of Z that uses $\mathrm{mcs}(Z)$ searchers and minimizes the number of exposed vertices at each step has the property that there exists $k_1 < k_2 < \cdots < k_n$ such that $\mathrm{ex}_S(Z, k_i) \geqq a_i$.*

Proof. Since Z contains an M-clique, we know that $\mathrm{s}(Z) \geqq M$. Further, there is a monotonic connected sweep strategy using M searchers. First, clear M, then the first $a_1 + 1$ clique, then the next, and so on, moving from left to right. Thus, $\mathrm{mcs}(Z) = M$.

Let v be the first vertex cleared in a monotonic connected sweep strategy S on Z using M searchers. If v is not in one of the M-cliques, then there is a cleared edge e in some other clique. There are at least two vertex-disjoint paths that pass through the vertices of e to either M. The first time that a vertex is cleared in either M-clique, there are $M - 1$ searchers on that clique. But at the same time, there are two vertex-disjoint paths from the vertices of e to the other M-clique. These two paths must contain at least one exposed vertex, and hence one searcher. But this sweep then uses $M + 1$ searchers. Thus, v must be in one of the M-cliques.

Let $i < j$. Consider a_i and a_j. Assume that no $a_i + 1$ clique obtains a cleared vertex before all the $a_j + 1$ cliques. Let w be a cleared vertex in one of the $a_j + 1$ cliques. Since S is a connected sweep strategy there is a cleared path between v and w. But this pass must pass through the $a_i + 1$ cliques, of which there are $M + 1$. Since these cliques contain no cleared vertices, there must be at least one exposed vertex in each of the $a_i + 1$ cliques, and hence at least $M + 1$ searchers in the $a_i + 1$ cliques. Since this uses too many searchers, some $a_i + 1$ clique must contain a cleared vertex before all the $a_j + 1$ cliques do. When the $a_i + 1$ clique first contains a cleared vertex, there are at least a_i exposed vertices. ■

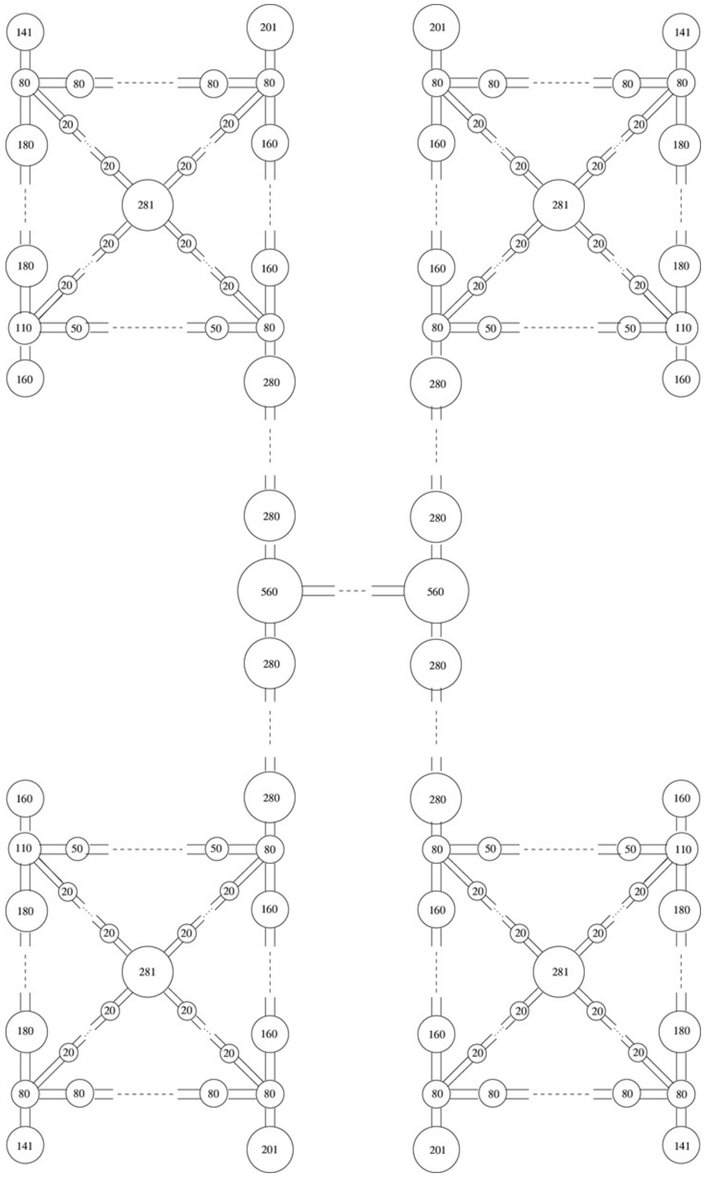

Fig. 6. The graph H.

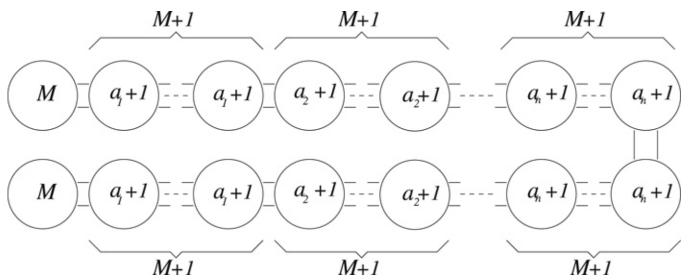

Fig. 7. The graph Z.

5. Conclusions

Constructing graphs with large cliques is useful, as these cliques imply lower bounds on the sweep number, they also allow us to restrict how a graph is cleared by setting up situations where ''paths'' of cliques must be cleared clique by clique rather than ''sneaking'' through them.

Having solved one of the open problems in [3], we are compelled to mention the other: find an upper bound for the ratio $\text{mcs}(G)/\text{s}(G)$. For the special case of trees, the bound of 2 was shown in [3], and the authors believe that it is true for all connected graphs. It has also been shown in [7] that $\text{cs}(G)/\text{s}(G) \leqq \log n + 1$ for any n-vertex G.

As mentioned in the introduction, the Y-square is the smallest known graph with sweep number strictly less than monotonic internal sweep number. For the other inequalities, we have given large examples of graphs which show that the inequalities can be strict. Smallest graphs with these properties would be interesting to find.

While constructing the W and demonstrating that $\text{cs}(W) < \text{mcs}(W)$, many additions had to be made to W to simplify the proof. Essentially, these were made so that any sweep strategy would go from the clique A_9' to the clique A_9, or vice versa. We pose the following problem relating to sweep structure: is there a graph G such that every monotonic connected sweep of G using $\text{mcs}(G)$ searchers must first clear an edge in the graph induced by vertex set U and last clear an edge in the graph induced by vertex set V, with $U \cap V = \emptyset$?

Acknowledgements

The authors would like to extend special thanks to Denis Hanson and Xiangwen Li both for their ongoing support and for the fascinating discussion they have provided in our weekly MITACS seminars. The first author's research was supported in part by NSERC under Grant 261290-03 and MITACS. The second author's research was supported in part by NSERC. The third author's research was supported in part by NSERC under Grant A-4792 and MITACS.

References

[1] B. Alspach, Searching and sweeping graphs: A brief survey, Le Matematiche (Catania) 59 (2004) 5–37.
[2] L. Barrière, P. Flocchini, P. Fraigniaud, N. Santoro, Capture of an intruder by mobile agents, in: Proc. of the 14th ACM Symposium on Parallel Algorithms and Architectures, SPAA 2002, 2002, pp. 200–209.
[3] L. Barrière, P. Fraigniaud, N. Santoro, D. Thilikos, Searching is not jumping, in: In Proc. 29th Workshop on Graph Theoretic Concepts in Computer Science, WG 2003, in: Lecture Notes in Computer Science, vol. 2880, 2003, pp. 34–45.
[4] D. Bienstock, P. Seymour, Monotonicity in graph searching, Journal of Algorithms 12 (1991) 239–245.
[5] R.L. Breisch, An intuitive approach to speleotopology, Southwestern Cavers 6 (1967) 72–78.
[6] M. Fellows, M. Langston, On search, decision and the efficiency of polynomial time algorithm, in: 21st ACM Symp. on Theory of Computing, STOC 89, 1989, pp. 501–512.
[7] P. Fraigniaud, N. Nisse, Connected treewidth and connected graph searching, in: Proceedings of 7th Latin American Theoretical Informatics Symposium (LATIN), in: Lecture Notes in Computer Science, vol. 3887, Springer-Verlag, Berlin, 2006, pp. 479–490.
[8] M. Frankling, Z. Galil, M. Yung, Eavesdropping games: A graph-theoretic approach to privacy in distributed systems, Journal of ACM 47 (2000) 225–243.
[9] L.M. Kirousis, C.H. Papadimitriou, Searching and pebbling, Theoret. Comput. Sci. 47 (2) (1986) 205–218.
[10] A.S. LaPaugh, Recontamination does not help to search a graph, Journal of ACM 40 (1993) 224–245.
[11] N. Megiddo, S.L. Hakimi, M. Garey, D. Johnson, C.H. Papadimitriou, The complexity of searching a graph, Journal of ACM 35 (1988) 18–44.
[12] T. Parsons, Pursuit-evasion in a graph, in: Theory and Applications of Graphs, in: Lecture Notes in Mathematics, Springer-Verlag, 1976, pp. 426–441.
[13] R. Tošić, The search number of the Cartesian product of graphs, Univ. Novom Sabu Zb. Rad. Prirod.-Mat Fak. Ser. Mat. 17 (1987) 239–243.
[14] B. Yang, D. Dyer, B. Alspach, Sweeping graphs with large clique number (extended abstract), in: Rudolf Fleischer, Gerhard Trippen (Eds.), Proceedings of 15th International Symposium on Algorithms and Computation, in: Lecture Notes in Computer Science, vol. 3341, Springer-Verlag, Berlin, 2004, pp. 908–920.