Recontamination Does Not Help to Search a Graph

ANDREA S. LAPAUGH

Princeton University, Princeton, New Jersey

Abstract. This paper is concerned with a game on graphs called *graph searching*. The object of this game is to clear all edges of a contaminated graph. Clearing is achieved by moving searchers, a kind of token, along the edges of the graph according to clearing rules Certain search strategies cause edges that have been cleared to become contaminated again. Megiddo et al. [9] conjectured that every graph can be searched using a minimum number of searchers without this recontamination occurring, that is, without clearing any edge twice. In this paper, this conjecture is proved. This places the graph-searching problem in NP, completing the proof by Megiddo et al. that the graph-searching problem is NP-complete. Furthermore, by eliminating the need to consider recontamination, this result simplifies the analysis of searcher requirements with respect to other properties of graphs.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Graph searching, NP-completeness, pursuit and evasion

1. *Introduction*

We are interested in a game on graphs that involves the clearing of contaminated edges. A graph is presented with all edges contaminated. Through a sequence of moves using *searchers*, we would like to obtain a state of the graph in which all edges are simultaneously clear. Searchers can be thought of as tokens that move around the graph. The object of the game is to use as few searchers as possible to reach the desired clear state. The allowable moves are as follows:

(1) Place a searcher on a node;
(2) Remove a searcher from a node;
(3) Move a searcher along an edge.

There are two ways in which a contaminated edge can become clear:

(1) There is a searcher on one endpoint of the edge. A second searcher is moved from that endpoint along the edge to the other endpoint;

This work was supported in part by AT & T Bell Laboratories, Murray Hill, N.J , and by National Science Foundation (NSF) grant MCS 82-02594.

Author's address: Princeton University, Computer Science Department, 35 Olden Street, Princeton, NJ 08544-2087 slp@princeton.edu

(2) There is a searcher on one endpoint of the edge, and all other edges incident on this node are clear. The searcher can be moved to the other endpoint of the edge, leaving all edges incident on the first endpoint clear.

A node that contains a searcher is called *guarded*. A path that contains a guarded node is also called *guarded*. An edge that is clear can become *recontaminated* if, after the edge is cleared, the graph reaches a state in which there is an unguarded path containing one endpoint of a contaminated edge and one endpoint of the cleared edge. We refer to this as a path from the contaminated edge to the cleared edge. The number of searchers used by a sequence of moves is defined to be the maximum number of searchers on the graph during the moves. A sequence of moves that takes a graph from the state in which all edges are contaminated to a state in which all edges are clear is called a *search strategy* for the graph.

The graph-searching problem was first discussed by Breisch [2]; early work was also done by Parsons [10, 11]. In [9], Megiddo et al. showed that the decision problem: "Given a graph G and a positive integer k, can G be cleared using no more than k searchers?" is NP-hard [4] but did not place it in NP. They conjectured that recontamination cannot help—that for every graph, there is a clearing strategy using a minimum number of searchers and under which no edge is ever recontaminated. This would show that the problem is in NP, since one could guess the sequence in which edges are cleared. In this paper, we prove that conjecture. This result completes the proof of NP-completeness for the decision problem.

The fact that recontamination is not necessary is not only useful for showing that the graph-searching problem is NP-complete. It is also useful in relating graph searching to other graph problems. The *search number* of a graph is the minimum number of searchers required by any search strategy that clears the graph. Kirousis and Papadimitriou [5] have defined a variant of graph searching, giving another measure called the *node search number*. Using our result that recontamination is not necessary, search number and node search number have been related to other important graph measures: cutwidth [6, 8], vertex separation [3, 5], interval thickness [5], and topological bandwidth [7].

2. *Transforming the Problem*

To prove this conjecture, we modify the game slightly by introducing a stronger version of recontamination. We prove that this stronger recontamination cannot help.

Define the new game as follows: The new game has two kinds of moves: recontamination moves and clearing moves. During a clearing move, exactly one edge is cleared. During a recontamination move, some nonempty set of edges is recontaminated. Let a node be called *clear* if all edges incident on it are clear, *contaminated* if all edges incident on it are contaminated, and *partially clear* if there are both contaminated and clear edges incident on it. There will be two conditions that are satisfied at the end of each move:

(I) No clear or contaminated node contains a searcher;
(II) No node contains more than one searcher.

Suppose we are given k searchers at the beginning of the game, and we wish to find a search strategy using no more than these k searchers. After any move, any of the k searchers that are not on the graph are called *free* searchers.

The moves are as follows:

Clearing move:

(1) Put down some number of searchers;
(2) Move a searcher along a contaminated edge to clear it according to the clearing rule of the original game;
(3) Pick up searchers that violate conditions (I) and (II).

Note that there are no unguarded paths from contaminated edges to clear edges after step (2), and step (3) cannot introduce any unguarded paths from contaminated to clear edges.

Recontamination move:

(1) Pick up some number of searchers (from partially clear nodes);
(2) Put down some number of searchers on unguarded nodes;
(3) Recontaminate any clear edge that lies on an unguarded path to a contaminated edge;
(4) Pick up searchers that violate conditions (I) and (II).

LEMMA 1. *For any search strategy under the old rules using k searchers, there is a search strategy under the new rules using k searchers.*

PROOF. Given any graph G and k searchers, we prove that, for each move under the old rules, there is a move under the new rules that results in the same set of contaminated edges. This is formally stated as follows:

Claim. Given a graph G, let s_o be a state of G that is legal under the old rules. (A state specifies which edges are contaminated and where searchers are located.) Let s_n be a state of G satisfying conditions (I) and (II) of the new rules such that:

(A) s_o and s_n have the same set of contaminated edges;
(B) the set of guarded nodes in s_n is a subset of the set of guarded nodes in s_o.

Then, for a given move under the old rules, m_o, applied to a state s_o, there is a move under the new rules, m_n, applied to s_n, such that (A) and (B) hold for the states resulting from these moves.

After proving this claim, Lemma 1 follows by induction on the number of moves in the search strategy under the old rules.

PROOF OF THE CLAIM. There are three cases.

(i) m_o is to place a searcher on a node. m_n is to do nothing. Since placing a searcher does not change the set of contaminated edges, (A) holds for the new states. Since the set of guarded nodes at most gains one new member due to m_o and is unchanged due to m_n, (B) must hold.

(ii) m_o is to remove a searcher from a node. If the node contains more than one searcher before m_o or is contaminated or clear, the set of contaminated edges will not change. In s_n, the node would not have a corresponding searcher. Therefore, (A) and (B) are satisfied if nothing is done under the new rules. If the node is partially clear and the only searcher is removed, recontamination will take place. In s_n, the graph will also contain a searcher since the node will be partially clear here, too. The following recontamination move under the new rules is carried out:

(1) pick up the searcher picked up in m_o;
(2) place a searcher on each node which contains one in s_o but not in s_n;
(3) recontaminate;
(4) pick up searchers which violate (I) or (II).

Step (2) can be done because the set of guarded nodes in s_n is a subset of that in s_o. Therefore, there are enough free searchers. Once step (2) is completed, the state of the graph for the new rules and state of the graph for the old rules just before propagating recontamination differ only in that the state under the old rules may have multiple searchers on each guarded node while under the new rules there is only one searcher. Therefore, after step (3) of m_n, the states of the graph again differ only as to the number of searchers on guarded nodes. After step (4), some nodes that are guarded under the old rules may not be guarded under the new rules. Thus, (A) and (B) hold.

(iii) m_o is to move a searcher along an edge, e. If this results in a new clear edge, then make the following clearing move under the new rules:

(1) Put down searchers on all nodes containing them in s_o;
(2) Move the searcher along e as done in the old move;
(3) Pick up searchers which violate (I) and (II).

As before, step (1) can be done since the set of guarded nodes in s_n is a subset of those containing searchers in s_o. After step (2), the state under the old rules and the state under the new rules differ only due to possible multiple searchers on a guarded node under the old rules. After step (3), some nodes that are guarded in the state under the old rules may not be guarded in the state under the new rules. Thus, (A) and (B) hold.

If moving the searcher in m_o does not result in clearing e, then the edge may or may not be recontaminated. If the searcher is on a clear or contaminated node in s_o, then no edges change their status due to the move. Under s_n, the searcher is not present. Therefore, even if the node becomes unguarded after m_o, the set of guarded nodes in s_n is a subset of the set of guarded nodes in the state resulting from m_o on s_o. Conditions (A) and (B) are satisfied without a move on s_n; move m_n is empty.

If the searcher is on a partially clear node in s_o, but there are other searchers on the node, then the searcher must move along a clear edge for no clearing to take place. No edges change their status due to this move. Under s_n, the graph contains one searcher on the partially clear node. Move m_o does not remove any node from the set of guarded nodes. Therefore, the set of guarded nodes in s_n is a subset of the set of guarded nodes in the state resulting from m_o on s_o. Move m_n is empty.

If the searcher to be moved is the only searcher on a partially clear node, then recontamination of some edge or edges will occur due to m_o on s_o. All edges incident on the node containing the searcher in s_o and all edges on unguarded paths from this node are recontaminated if originally clear. The moved searcher may serve to guard paths through its new node. The move m_n is:

(1) pick up the searcher moved in m_o;
(2) put this searcher on the other endpoint of e. Put a searcher on any node guarded in s_o but not in s_n, excluding the endpoints of e;
(3) recontaminate;
(4) pick up searchers which violate (I) and (II).

Step (2) is feasible, since at this point the set of guarded nodes other than the endpoints of e in s_n is a subset of the set of guarded nodes other than the endpoints of e in s_o. The state of the graph after step (2) and the state of the graph during m_o after the searcher is moved but without having propagated recontamination differ only by multiple searchers on guarded nodes under the old rules. Therefore, the same edges are recontaminated under m_n and m_o. Step (4) leaves the set of guarded nodes a subset of those guarded in the state resulting from m_o on s_o. Thus, (A) and (B) hold after m_o is performed on s_o, and m_n is performed on s_n. This is the last case to consider, and the claim is proven. □

LEMMA 2. *For any search strategy under the new rules that uses k searchers and contains no recontamination moves, there is a search strategy under the old rules that uses k searchers and has no recontamination.*

PROOF. The clearing move under the new rules is composed of legal moves under the old rules. None of these can lead to recontamination under the old rules. □

The remainder of this paper is concerned with proving the following theorem:

THEOREM 1. *If there is a search strategy for a graph under the new rules using k searchers, then there is a search strategy for the graph without any recontamination moves and still using k searchers.*

This theorem allows us to obtain the main theorem of this paper:

THEOREM 2. *If there is a search strategy for a graph under the old rules using k searchers, there is a search strategy for the graph without recontamination using k searchers.*

PROOF. By Lemma 1, if there is a search strategy for a graph under the old rules using k searchers, there is a search strategy under the new rules using k searchers. By Theorem 1, there is a search strategy for the graph under the new rules using k searchers that does not use any recontamination moves. Finally, given this strategy, which does not have recontamination, Lemma 2 guarantees that there is a search strategy for the graph under the old rules that uses k searchers and that does not have recontamination. □

3. *Proof of Theorem* 1: *Manipulating Recontamination Moves*

The remainder of this paper deals only with search strategies under the new rules. Our method of proving Theorem 1 depends on four ways in which we are able to manipulate recontamination moves. We present these informally now, postponing for the moment precise definitions of some terms (such as *certain small separating set*):

(1) We can combine any two consecutive recontamination moves (Lemma 4);
(2) We can decompose a recontamination move into two recontamination moves if there is a certain small separating set (Lemma 5);
(3) If a clearing move is followed by a recontamination move that does not consume free searchers, then the clearing move can be postponed until after the recontamination move (Lemma 6);
(4) If (1), (2), and (3) above do not apply to the very last recontamination move in a search strategy, then this recontamination move can be eliminated (Lemma 7).

Using these four manipulations, we are able to either reduce the number of clearing moves before the last recontamination move in a search strategy or reduce the number of edges recontaminated in the last recontamination move (Lemma 8). Repeated application of this reduction will eventually eliminate all recontamination moves, proving Theorem 1.

We now proceed to develop formally the proof of Theorem 1 outlined above. We begin by presenting some notation that will be used throughout this section. Given a recontamination move, r, let R denote the set of nodes from which searchers are removed in step (1), and let P denote the set of nodes on which searchers are placed in step (2). Let N_{gr} denote the set of guarded nodes before the recontamination move and $N_{\mathrm{gr}}(r)$ denote the set of guarded nodes after the recontamination move. Similarly, let E_{cntm} denote the set of contaminated edges before the recontamination move and $E_{\mathrm{cntm}}(r)$ denote the set of contaminated edges after the recontamination move. If we are considering a sequence of consecutive (recontamination or clearing) moves $m_1, m_2, \ldots, m_k$, then $N_{\mathrm{gr}}(m_1 m_2 \cdots m_i)$ is the set of guarded nodes and $E_{\mathrm{cntm}}(m_1 m_2 \cdots m_i)$ is the set of contaminated edges after the first i moves have been executed. $N_{\mathrm{gr}}(m_1 m_2 \cdots m_i)$ and $E_{\mathrm{cntm}}(m_1 m_2 \cdots m_i)$ specify the state of the graph after the first i moves. However, note that because of conditions (I) and (II), $E_{\mathrm{cntm}}(m_1 m_2 \cdots m_i)$ is sufficient to specify the state—exactly the partially clear nodes contain searchers.

It will be convenient for many of the proofs to deal with a more "well-behaved" version of the recontamination move:

Definition. A recontamination move is *irredundant* if

(i) R and P are disjoint;
(ii) Step (4) is unnecessary, that is, no searchers are removed in step (4).

LEMMA 3. *Any recontamination move can be replaced by an irredundant recontamination move that leaves the graph in exactly the same state.*

PROOF. If R and P are not disjoint, the irredundant move does not touch searchers on the nodes in $P \cap R$; the set of guarded nodes will be unchanged at the beginning of step (3). Step (4) can be eliminated by noting that only condition (I) can become violated—extra searchers are never added to a guarded node. If a contaminated node contains a searcher after step (3), the searcher can be picked up at step (1) instead or never put down in step (2) without affecting the set of edges recontaminated. To see this, note that every path from the guarded contaminated node to a clear edge must be guarded by another searcher. If a clear node contains a searcher after step (3), this searcher must have been placed in step (2). This searcher does not guard any path from a contaminated edge to a clear edge that is not also guarded elsewhere (or some edge incident on the node would become recontaminated). Therefore, this searcher needs not be placed in step (2). □

The following simple properties of an irredundant recontamination move, r, will also be useful. (Our numbering of properties includes both the definitional properties above and these derivative properties so that we may uniformly refer to properties (i)–(v) of an irredundant recontamination move.)

(iii) $N_{\text{gr}}(r) = (N_{\text{gr}} - R) \cup P$;
(iv) Any node in R is partially clear before r and contaminated after r;
(v) Any node in P is clear before r and partially clear after r.

Properties (i)–(v) of an irredundant recontamination move will simplify several of the proofs in this section. However, it will sometimes be simpler to prove a lemma if recontamination moves that are not irredundant are allowed. Therefore, we always state when we are considering irredundant recontamination moves.

We now begin the development of our manipulation capabilities. We begin with combining moves. Combining moves will allow us to view a strategy as having at most one recontamination move between any two clearing moves.

LEMMA 4. *Any two recontamination moves that are performed without clearing an edge in between can be combined into one recontamination move such that the graph will be in the same state after the combined move as after the two consecutive moves.*

PROOF. Denote the original recontamination moves as r_1 and r_2, in order of their execution. Without loss of generality, assume that these moves are irredundant. For $i = 1$ or 2, move r_i removes searcher from R_i and places searchers on P_i. The combined recontamination move, denoted r, is as follows:

(1) Pick up searchers from nodes in $R = R_1 \cup [R_2 - P_1]$;
(2) Place searchers on nodes in $P = P_2 \cup [P_1 - R_2]$;
(3) Recontaminate.

We must show that each node in $N_{\text{gr}}(r)$ contains exactly one searcher and that $N_{\text{gr}}(r) = N_{\text{gr}}(r_1 r_2)$. This will prove that r does not require more searchers than r_1 or r_2, and thus r is legal. To prove that the graph is in the same state after r as after the sequence $r_1 r_2$, we must also show that $E_{\text{cntm}}(r) = E_{\text{cntm}}(r_1 r_2)$. Once we have shown that the same edges have been recontaminated, this will also prove that step (4) is not needed in r since no clear or contaminated nodes are guarded after r_2.

We begin by proving that each node contains at most one searcher after r. First note that R_1 and P_2 are disjoint since after r_1 the nodes in R_1 are contaminated (property (iv)), and we do not place searchers on contaminated nodes. Given this and that P_1 and R_1 are disjoint and P_2 and R_2 are disjoint (property (i)), we can conclude that R and P are disjoint. For a node to contain two searchers after r, violating condition (II), the node must originally contain a searcher, be in P, and (since r_1 and r_2 satisfy condition (II)) be in R_1, that is, the node must be in $N_{\text{gr}} \cap P \cap R_1$. But R_1 is disjoint from P_1 and P_2 and thus P, so this is not possible, and r satisfies condition (II).

To show that $N_{\mathrm{gr}}(r) = N_{\mathrm{gr}}(r_1 r_2)$, we first expand the definition of $N_{\mathrm{gr}}(r)$:

$$\begin{aligned} N_{\mathrm{gr}}(r) &= \big(N_{\mathrm{gr}} - (R_1 \cup (R_2 - P_1))\big) \cup (P_2 \cup (P_1 - R_2)) \\ &= \Big(\big(N_{\mathrm{gr}} - (R_2 - P_1)\big) - R_1\Big) \cup (P_1 - R_2) \cup P_2. \end{aligned}$$

Since N_{gr} and P_1 are disjoint:

$$N_{\mathrm{gr}} - (R_2 - P_1) = N_{\mathrm{gr}} - R_2$$

and

$$N_{\mathrm{gr}}(r) = \big(N_{\mathrm{gr}} - (R_1 \cup R_2)\big) \cup P_2 \cup (P_1 - R_2).$$

Now applying the definition of r_1 and r_2 we have:

$$\begin{aligned} N_{\mathrm{gr}}(r_1 r_2) &= \big(N_{\mathrm{gr}}(r_1) - R_2\big) \cup P_2 \\ &= \Big(\big((N_{\mathrm{gr}} - R_1) \cup P_1\big) - R_2\Big) \cup P_2 \\ &= \big(N_{\mathrm{gr}} - (R_1 \cup R_2)\big) \cup (P_1 - R_2) \cup P_2 \end{aligned}$$

and $N_{\mathrm{gr}}(r) = N_{\mathrm{gr}}(r_1 r_2)$ as desired.

It remains to show that $E_{\mathrm{cntm}}(r) = E_{\mathrm{cntm}}(r_1 r_2)$. Since no edges are cleared in any of the moves, we need only worry about recontamination. We first prove $E_{\mathrm{cntm}}(r_1 r_2) \subseteq E_{\mathrm{cntm}}(r)$. Suppose an edge is clear initially and contaminated after r_2. Then there are two cases: it is either recontaminated during r_1 or during r_2.

Case 1. Suppose it is recontaminated during r_1. Then during r_1, there is an unguarded path to this edge from an edge in E_{cntm}. In the combined move, this path is guarded only if a searcher was placed on the path during r_2, and therefore also placed in the combined move. However, after r_1, every node along the path is contaminated, so no searchers would be placed on the path during r_2. Therefore, the path is unguarded in the combined move, and the edge in question is in $E_{\mathrm{cntm}}(r)$.

Case 2. Suppose recontamination occurs during r_2. Then there is an unguarded path during r_2 from a contaminated edge, e, to the edge being recontaminated. Since $N_{\mathrm{gr}}(r) = N_{\mathrm{gr}}(r_1 r_2)$, this path must be unguarded in the combined move. Edge e is in E_{cntm} or $E_{\mathrm{cntm}}(r_1) - E_{\mathrm{cntm}}$. We have just shown in Case 1, that for any edge in $E_{\mathrm{cntm}}(r_1) - E_{\mathrm{cntm}}$, there is an unguarded path during the combined move from a contaminated edge to this edge. The composition of this path and the unguarded path of r_2 gives an unguarded path in the combined move from a contaminated edge to the edge recontaminated in r_2. Therefore, this edge is also recontaminated in the combined move.

We have just proven that $E_{\mathrm{cntm}}(r_1 r_2) \subseteq E_{\mathrm{cntm}}(r)$. Now we prove $E_{\mathrm{cntm}}(r) \subseteq E_{\mathrm{cntm}}(r_1 r_2)$. For any edge contaminated during r, there must be an unguarded path from the edge to an edge in E_{cntm}. If the edge is not recontaminated in r_1, it is because there is some guarded node along this path. Consider the guarded node nearest to the edge. This node must be partially clear after r_1. However, since $N_{\mathrm{gr}}(r) = N_{\mathrm{gr}}(r_1 r_2)$, the searcher must be removed from this node in r_2. Therefore, during r_2, there is an unguarded path from the edge in question to a contaminated edge; the edge must be recontaminated during r_2. This completes the proof that $E_{\mathrm{cntm}}(r) = E_{\mathrm{cntm}}(r_1 r_2)$ and the proof of the lemma. □

We now consider decomposing a recontamination move. The motivation for this decomposition is as follows: Intuitively, there are good recontamination moves (those that leave us with more free searchers) and bad recontamination moves (those which leave us with fewer free searchers). Our methods for handling these two types will be different. However, a seemingly bad recontamination move may be a combination of a good move and a bad one. Lemma 5 shows how to decompose certain recontamination moves so that the first of the two resulting moves gives more free searchers. The decomposition is based on finding a small separating set of nodes which can be used to allow only part of the original recontamination.

Definition. For a given irredundant recontamination move, r, with searchers removed from R and placed on P, a set of nodes C is an *unguarded separating set* of R and P if C is a separating set for R and P in $G - (N_{\text{gr}} - R)$,[1] that is, every path from a node in R to a node in P contains a node of C or a node of $N_{\text{gr}} - R$. Set C need not be disjoint from R or P but is disjoint from $N_{\text{gr}} - R$. C is not properly contained in P. (Suppose it were. Each node v in $P - C$ is clear before r and partially clear after r. Therefore, during recontamination there must be an unguarded path to v from some node in R. But such a path must contain an element of C. Since $C \subset P$, this element of C is guarded and cannot be in R. Thus, the path is guarded at an interior node, a contradiction.)

LEMMA 5. *Given a graph G in an arbitrary state of contamination, let m be an irredundant recontamination move with searchers removed from R and placed on P. If there is a minimal unguarded separating set, C, separating R and P such that $|C| < |R|$ and $C \neq P$, then the recontamination move can be decomposed into two recontamination moves. The first move of the decomposition, m_1, has searchers removed from $R_1 = R - C$ and placed on $P_1 = C - R$. The second move, m_2, has searchers removed from $R_2 = C - P$ and placed on $P_2 = P - C$. Both moves are irredundant.*

PROOF. Note that if P is empty, the minimal unguarded separating set is also empty. Thus, we only deal with $P \neq \phi$. We must show that m_1 and m_2 are both legal and irredundant and that the state of the graph after the sequence $m_1 m_2$ is the same as that after m. To show that m_i is legal, for $i = 1$ or 2, we must show that nodes in R_i are guarded at the beginning of step (1) of m_i, that nodes in P_i are unguarded at the beginning of step (2) of m_i, and that there are enough searchers to execute the move. We begin by considering legality and irredundancy, and then consider the state of the graph. However, we postpone proving property (ii) of irredundancy—that step (4) is unnecessary—until last, since it will use the fact that the same edges are recontaminated after $m_1 m_2$ as after m.

Legality and irredundancy of m_1. Clearly $R_1 = R - C$ and $P_1 = C - R$ are disjoint (property (i) of irredundancy). Since $R - C \subseteq R$, these nodes are guarded initially; since C is disjoint from $N_{\text{gr}} - R$, nodes in $C - R$ are unguarded initially. Since $|C| < |R|, |C - R| < |R - C|$ and m_1 can be done, even if all allowed searchers are in use at the beginning of the move. We

[1]$G - (N_{\text{gr}} - R)$ is the subgraph of G formed by removing all guarded nodes not in R.

postpone proving property (ii) of irredundancy until we have shown that the state of the graph after $m_1 m_2$ is the same as that after m.

Legality and irredundancy of m_2. Clearly $R_2 = C - P$ and $P_2 = P - C$ are disjoint (property (i) of irredundancy). After m_1, all nodes in C are guarded, and in particular, the nodes in R_2 are guarded. Since $N_{\text{gr}}(m_1) \subseteq N_{\text{gr}} \cup C$ and N_{gr} and P are disjoint, nodes in $P - C = P_2$ are unguarded after m_1. The number of free searchers after m_1 is increased by

$$|R - C| - |C - R| = |R| - |R \cap C| - (|C| - |R \cap C|) = |R| - |C| > 0$$

from the number before the move. If $|P| \geq |C|$, $|P - C| - |C - P| = |P| - |C|$ of these searchers are needed for m_2. If $|P| \geq |R|$, then in the original move m, $|P| - |R|$ free searchers were used. These, plus the $|R| - |C|$ extra free searchers provide enough searchers for m_2. Property (ii) of irredundancy for m_2 will follow immediately from the graph being in the same state after $m_1 m_2$ as after m.

State of the graph. To show that the graph is in the same state after $m_1 m_2$ as after m, we must show that $N_{\text{gr}}(m) = N_{\text{gr}}(m_1 m_2)$ and that $E_{\text{cntm}}(m) = E_{\text{cntm}}(m_1 m_2)$. Consider first $N_{\text{gr}}(m_1 m_2)$:

$$\begin{aligned}
N_{\text{gr}}(m_1 m_2) &= \big(N_{\text{gr}}(m_1) - R_2\big) \cup P_2 \\
&= \Big(\big((N_{\text{gr}} - R_1) \cup P_1\big) - R_2\Big) \cup P_2 \\
&= \Big(\big((N_{\text{gr}} - (R - C)) \cup (C - R)\big) - (C - P)\Big) \cup (P - C) \\
&= \big((N_{\text{gr}} - (R - C)) - (C - P)\big) \\
&\quad \cup ((C - R) - (C - P)) \cup (P - C) \\
&= \big(N_{\text{gr}} - ((R - C) \cup (C - P))\big) \\
&\quad \cup ((C - R) - (C - P)) \cup (P - C).
\end{aligned}$$

The following identities will allow us to simplify this expression for $N_{\text{gr}}(m_1 m_2)$: Since P and R are disjoint,

$$(C - R) - (C - P) = (C - (C - P)) - R = (C \cap P) - R = C \cap P$$

and

$$R - C = R - (C - P).$$

Therefore

$$(R - C) \cup (C - P) = (R - (C - P)) \cup (C - P) = R \cup (C - P).$$

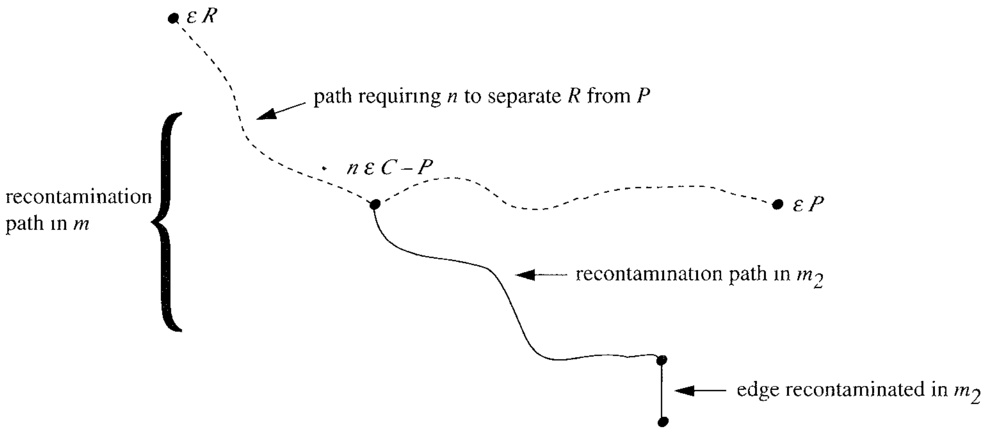
Furthermore, since C and $N_{\text{gr}} - R$ are disjoint,

$$\begin{aligned}
N_{\text{gr}} - ((R - C) \cup (C - P)) &= N_{\text{gr}} - (R \cup (C - P)) \\
&= (N_{\text{gr}} - R) - (C - P) = N_{\text{gr}} - R.
\end{aligned}$$

Finally, we have

$$N_{\text{gr}}(m_1 m_2) = (N_{\text{gr}} - R) \cup (C \cap P) \cup (P - C) = (N_{\text{gr}} - R) \cup P = N_{\text{gr}}(m),$$

as desired.



εR
path requiring n to separate R from P
$n \varepsilon C-P$
εP
recontamination path in m
recontamination path in m_2
edge recontaminated in m_2

FIG. 1. Recontamination during m_2 implies recontamination during m.

We now prove that $E_{\text{cntm}}(m) = E_{\text{cntm}}(m_1m_2)$. We first show that $E_{\text{cntm}}(m) \subseteq E_{\text{cntm}}(m_1m_2)$. Suppose an edge is recontaminated during m. Then there is an unguarded path from a node in R to this edge after the move. This path must also be unguarded after m_2. Therefore, the edge must be contaminated after m_2 and $E_{\text{cntm}}(m) \subseteq E_{\text{cntm}}(m_1m_2)$.

We now prove $E_{\text{cntm}}(m_1m_2) \subseteq E_{\text{cntm}}(m)$. First, suppose an edge is recontaminated during m_1. Then, there is an unguarded path from a node in $R - C$ to it after m_1. This path cannot contain a node in P, because every path from a node in R to a node in P goes through a node in C or in $N_{\text{gr}} - R$, but all nodes in C and $N_{\text{gr}} - R$ are guarded after m_1. Therefore, the path is also unguarded after m, and the edge is also recontaminated during m.

Now suppose an edge is recontaminated during m_2. Then there is an unguarded path after m_2 from a node, n, in $C - P$ to the edge. This path cannot contain a node in P, since all nodes in P are guarded after m_2. There must be a path from some node (possibly n) in R to n, which contains no node in P and no other node in C or $N_{\text{gr}} - R$ (see Figure 1). Otherwise, n could be removed from C to get a smaller unguarded separating set, and C would not be minimal. Combining this path from a node in R to n and the path from n to the edge recontaminated during m_2 gives a path that contains no node in $N_{\text{gr}} - R$ or P from a node in R to the edge. This path is unguarded after m, and the edge is recontaminated during m. This completes the proof that $E_{\text{cntm}}(m_1m_2) \subseteq E_{\text{cntm}}(m)$.

We have just shown that $E_{\text{cntm}}(m) = E_{\text{cntm}}(m_1m_2)$. With $N_{\text{gr}}(m) = N_{\text{gr}}(m_1m_2)$, this proves that the state of the graph after m_1m_2 is the same as that after m. To complete the proof of Lemma 5, we must show that step (4) of m_1 has been correctly omitted, that is, that every guarded node after m_1 is partially clear. (We know that step (4) of m_2 has been correctly omitted since we have just proven above that the state after this move is the same as the state after m.) A node, g, which is guarded after m_1, is either an element of $N_{\text{gr}} - R$ or an element of C. If the node is an element of $N_{\text{gr}} - R$, then it is partially clear before m_1 and after m_2. Therefore, it must be partially clear after m_1, since no clearing move separates the two recontamination moves.

If g is an element of C, we first show that g is partially clear or contaminated after m_1, that is, that it cannot be clear, and then that g is in fact partially clear. Node g lies on a path p from a node in R to a node in P that contains no nodes in R or P other than its endpoints and no node in $N_{gr} - R$ or C other than g (which may be an endpoint). Otherwise, C would not be minimal. If g is in R, then g is partially clear before m_1 and either partially clear or contaminated after m_1. If g is not in R, then the subpath from R to g must be from a node in $R - C$ and must consist of contaminated edges after m_1. Again, g is either partially clear or contaminated after m_1. Thus, in no case is g clear after m_1. If g is in P as well as C, then it is also guarded after m_2 and is therefore partially clear after m_2 and after m_1. Suppose g is in $C - P$. If every edge incident on it is contaminated after m_1, then every edge on the subpath of p from it to P must be contaminated since no node on this path except g is guarded. But this implies that the node in P is contaminated after m_1. This is impossible, since the node is partially clear after m_2. Therefore, at least one edge incident on g is clear, so g is partially clear. □

We now present the technical lemmas that let us deal with the two types of recontamination moves: those that leave at least as many searchers free after the recontamination move (good moves) and those that leave less searchers free after the recontamination move (bad moves). Without loss of generality, we consider only connected graphs with more than two nodes. (Connected components can be searched independently. The graph consisting of one edge, while trivial to search, adds unnecessary complexity to the case analyses below.) Lemma 7 allows us to eliminate bad recontamination moves that cannot be decomposed under Lemma 5. We are not able to eliminate good recontamination moves directly. However, Lemma 6 shows that if a good recontamination move is preceded by a clearing move, the clearing move can be postponed. Of course, if we can postpone all the clearing moves until after all the recontamination moves, we will have eliminated the recontamination moves.

LEMMA 6. *Let s_r be a recontamination move that is immediately preceded by a clearing move s_c in a search strategy for a given graph, G. If the number of free searchers after s_r is at least as large as the number before s_r, then moves s_c and s_r can be replaced by a new recontamination move t_r and new clearing move t_c such that t_r precedes t_c and the state of the graph after the sequence of moves $t_r t_c$ is the same as the state of the graph after $s_c s_r$. One or both of t_r and t_c may be empty moves; if t_c is not empty, it clears the same edge as s_c.*

PROOF. If the sequence $s_c s_r$ changes the state of the graph, then at least one of t_r and t_c will be nonempty. Let e be the edge cleared in s_c. Move t_r will recontaminate the same set of edges as s_r except for e; t_c will either be empty or clear e. The definitions of t_r and t_c will depend on whether or not e is clear after s_r.

Without loss of generality, we assume that s_r is irredundant. Let R_s be the nodes in step (1) of s_r and P_s be the nodes in step (2) of s_r. Since there are at least as many free searchers after s_r as before the move ($|R_s| \geq |P_s|$), all the searchers placed in step (2) can come from those removed in step (1). We use this fact often in modifying s_r to obtain t_r. Let m_e and n_e be the nodes that are the endpoints of e. To determine the specification of t_r and t_c, we now consider the two cases defined by the state of e after s_r.

Case I. e is recontaminated during s_r. In this case, t_c will be empty since e is contaminated after $s_c s_r$. We need only define t_r:

(1) Pick up searchers from nodes in $R_t = (R_s - \{m_e, n_e\}) \cup ((\{m_e, n_e\} \cap N_{gr}) - N_{gr}(s_c s_r))$;
(2) Place searchers on nodes in $P_t = P_s - \{m_e, n_e\}$;
(3) Recontaminate.

We must show that t_r is legal and that the state of the graph after t_r is the same as that after $s_c s_r$, that is, that $N_{gr}(t_r) = N_{gr}(s_c s_r)$ and $E_{cntm}(t_r) = E_{cntm}(s_c s_r)$. Once we have established that $N_{gr}(t_r) = N_{gr}(s_c s_r)$ and $E_{cntm}(t_r) = E_{cntm}(s_c s_r)$, we can conclude that step (4) has been correctly eliminated from the definition of t_r.

We begin by considering the legality of t_r. Sets N_{gr} and $N_{gr}(s_c)$ can only disagree on m_e and n_e. Therefore, all the nodes in $P_s - \{m_e, n_e\} = P_t$ are clear before s_c (equivalently before t_r), and t_r correctly places searchers only on clear nodes. The nodes in R_t other than m_e and n_e are in R_s and are guarded before t_r. Move t_r only removes a searcher from m_e or n_e if it is in N_{gr}. Therefore, t_r correctly removes searchers only from guarded nodes. To complete the proof that t_r is legal, we must show that there are enough searchers; this requires more careful analysis of the states of m_e and n_e after s_r and is postponed to Subcases (I.1) and (I.2) below.

We now consider the proof of $N_{gr}(t_r) = N_{gr}(s_c s_r)$. Sets R_t and R_s and sets P_t and P_s as well as sets N_{gr} and $N_{gr}(s_c)$ agree on all nodes other than the endpoints of e. Therefore, since $N_{gr}(t_r) = (N_{gr} - R_t) \cup P_t$ and $N_{gr}(s_c s_r) = (N_{gr}(s_c) - R_s) \cup P_s$, we need only show that $N_{gr}(t_r)$ and $N_{gr}(s_c s_r)$ agree on m_e and n_e. Again this will require the more careful analysis presented in cases (I.1) and (I.2).

Once we have completed the proofs that t_r is legal and $N_{gr}(t_r) = N_{gr}(s_c s_r)$, the equality of $E_{cntm}(t_r)$ and $E_{cntm}(s_c s_r)$ can be shown as follows. If $N_{gr}(t_r) = N_{gr}(s_c s_r)$, the set of guarded nodes during the recontamination step of t_r is the same as during that of s_r. Any difference in the recontamination during t_r and s_r will be due to e. Since e is recontaminated during s_r, the fact that e starts out contaminated in t_r will not cause added edges to be recontaminated. Therefore, we are able to conclude that $E_{cntm}(t_r) = E_{cntm}(s_c s_r)$. Thus, no clearing move t_c is necessary.

We now turn to our subcases. Note that since e is recontaminated during s_r, one endpoint of e, say n_e, must be of degree at least 2 and unguarded after s_r. Our subcases are defined by the state of the other endpoint, m_e, and after s_r.

Subcase I.1. After s_r, neither endpoint of e is guarded. Let δ_1 be the number of searchers on endpoints of e before s_c and δ_2 be the number of searchers on endpoints of e after s_c. Each of t_r and s_r remove any searchers on the endpoints of e before recontamination—δ_1 are removed by t_r and δ_2 by s_r. Thus, the states of these endpoints agree during and after the recontamination steps. $|R_t| = |R_s| + (\delta_1 - \delta_2)$. If $\delta_1 < \delta_2$, $|R_t|$ may be less than $|P_t|$. However, in this case there are $\delta_2 - \delta_1$ extra free searchers before s_c compared with after s_c. These free searchers are used to supplement the searchers from nodes in R_t to place searchers on P_t. Thus, t_r is legal. Note that t_r need

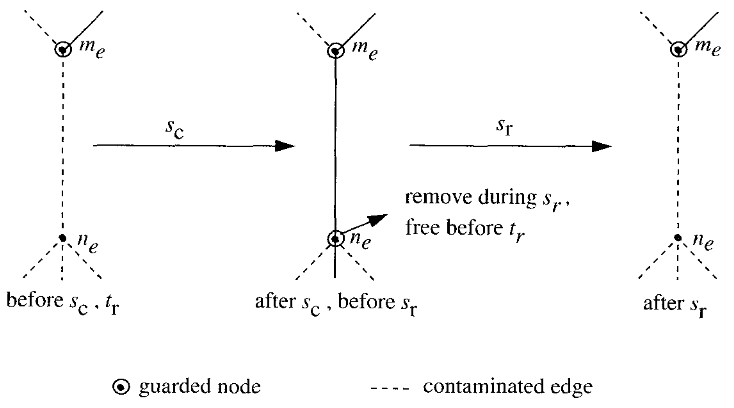


FIG. 2. t_r uses an extra free searcher.

not have the property that the number of free searchers after recontamination is at least as large as before recontamination.

Subcase I.2. After s_r, m_e is guarded. Since m_e is guarded after and thus during s_r, e must be recontaminated through n_e. After s_r, n_e is contaminated and some edge incident on m_e other than e is clear. This edge must be clear before s_c; therefore, m_e is also guarded before s_c. t_r leaves m_e guarded but removes any searcher on n_e before recontamination. Thus, the states of the endpoints of e agree during recontamination in t_r and s_r.

Suppose $|R_t| < |P_t|$. Then, n_e must be in R_s but not R_t since otherwise $|R_t|$ would be at least as large as $|R_s|$ while $|P_t|$ is no larger than $|P_s|$ ($|R_t| \geq |R_s| \geq |P_s| \geq |P_t|$). Also, m_e must be guarded after s_c. Otherwise, P_s would contain m_e and $|P_t|$ would equal $|P_s| - 1$, giving $|R_t| = |R_s| - 1 \geq |P_s| - 1 = |P_t|$. Thus, we are left with the case in which only m_e is guarded before s_c and both m_e and n_e are guarded after s_c (see Figure 2). Therefore, there is one extra free searcher before s_c in comparison with after s_c (i.e., before s_r). This extra searcher is used to supplement the searchers from $|R_t|$. We conclude that the number of available searchers for placement during t_r is at least $|R_t| + 1 = |R_s| \geq |P_s| = |P_t|$. Therefore, t_r is legal.

Case II. e is clear after s_r. Then R_s contains neither endpoint since removing a searcher from an endpoint of e would lead to its recontamination. Note that in the definition of t_r below, step (4) may remove searchers from m_e and/or n_e if they become contaminated during recontamination. This step can always be eliminated, but the proof is simpler if this is not done in the definition of t_r.

Move t_r:

(1) Pick up searchers from nodes in $R_t = R_s$;
(2) Place searchers on nodes in $P_t = P_s - \{m_e, n_e\}$;
(3) Recontaminate;
(4) Pick up searchers that violate conditions (I) and (II).

Move t_r is obviously legal. However, we must show that the same set of edges are recontaminated by t_r and s_r and define t_c to clear e. (Recall that $E_{\text{cntm}}(t_r)$ is sufficient to specify the state of the graph after t_r since conditions (I) and (II) are guaranteed by step (4).)

Let us first consider the recontamination that occurs during t_r and s_r. Note that during the recontamination steps of s_r and t_r, the only nodes that may be guarded in one and not the other are the endpoints of e. If an unguarded path from a contaminated edge to a clear edge during s_r is guarded during t_r, it must contain an endpoint of e. But this implies that e is recontaminated during s_r, which cannot be. Conversely, suppose there is an unguarded path from a contaminated edge to a clear edge during t_r, which is guarded during s_r. Such a path must contain an endpoint of e, which is unguarded (and therefore contaminated) before s_c and of degree ≥ 2. Without loss of generality, let m_e be the closest such endpoint to the clear edge; then, e does not appear on the subpath from m_e to the clear edge. Since m_e must be contaminated before s_c, all edges except e incident on m_e are contaminated before s_r (see Figure 3). Therefore, the first edge on the subpath from m_e to the clear edge is contaminated during s_r. Since this path is unguarded during s_r, the edge is recontaminated in s_r as well as t_r. We concluded that the same set of edges is recontaminated during s_r and t_r. After these moves, the states of all edges except e agree.

Let us now consider t_c to clear e. If after t_r, edge e can be cleared without using a free searcher (Figure 4(a)), then t_c is this clearing move. After t_c, the graph is in the same state as after $s_c s_r$.

Suppose that after t_r edge e requires two free searchers to be cleared (Figure 4(b)). Then both endpoints of e are contaminated and of degree ≥ 2. After s_r, both these endpoints must be guarded, since e is the only clear edge adjacent to either endpoint. Therefore, there are two searchers on the graph after s_r that are not on the graph after t_r; these searchers are free to be used by t_c.

Finally, suppose after t_r edge e requires one free searcher to be cleared. If s_c also requires a free searcher (Figure 4(c)), then this searcher is free after t_r since s_c and t_r begin with the same state of the graph and t_r removes at least as many searchers as it places. t_c uses this free searcher to clear e. If s_c does not require a free searcher (Figure 4(d)), then s_c clears e by moving a searcher that is already on one endpoint of e, say n_e; n_e is clear after s_c. Since t_c requires a free searcher, some edge adjacent to n_e must be recontaminated during t_r; this edge is also recontaminated during s_r. But then $n_e \in P_s$ or e would be recontaminated during s_r. Therefore, $|P_t| < |P_s|$. Since $|R_t| = |R_s| \geq |P_s| > |P_t|$, there is at least one free searcher after t_r. t_c uses this free searcher to clear e.

We have shown that in all cases, the state of the graph after $t_r t_c$ is the same as that after $s_c s_r$. This concludes the proof for Lemma 6. □

LEMMA 7. *Suppose s_r is the last recontamination move in a search strategy S for a graph G and is immediately preceded by a clearing move s_c. Assume s_r is irredundant. If the number of free searchers after s_r is less than the number before s_r and the sets R and P defined in steps* (1) *and* (2) *of s_r have no unguarded separating set C for which $|C| < |R|$, then there is another search strategy Γ for G using the same number of searchers with the following properties. The sequences of*

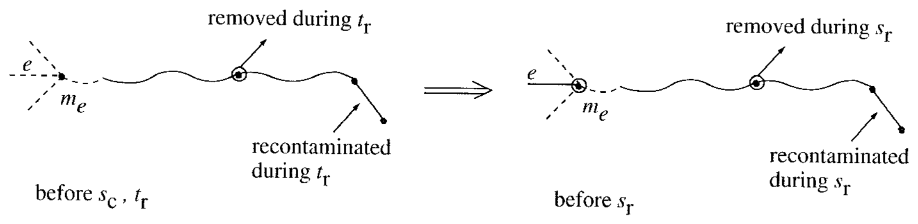


FIG. 3. Recontamination during t_r implies recontamination during s_r.

moves in Γ *and* S *are identical through* s_c; Γ *contains no recontamination moves following* s_c, *that is,* s_r *is eliminated. Thus,* Γ *has one fewer recontamination move than* S *while using the same number of searchers.*

PROOF. Γ is specified through s_c. We must describe the remaining sequence of moves for Γ, all of which are clearing moves. In S, all moves following s_r are clearing moves. For each such clearing move that clears an edge that was contaminated before s_r, there will be a corresponding clearing move in Γ clearing the same edge. The remaining clearing moves in S clear edges that are recontaminated during s_r. These edges will be clear in Γ after s_c; no corresponding clearing moves are necessary. The sequence of edges cleared in Γ after s_c is a subsequence of the sequence of edges cleared in S after s_c and s_r. Each move in Γ corresponds to a move in S, and we refer to *the move in* Γ *corresponding to a move in* S.

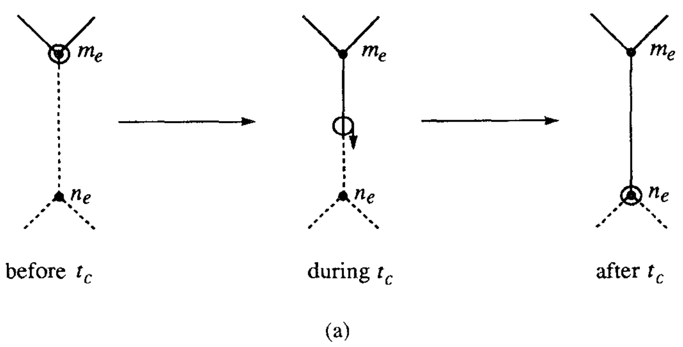
To prove that the clearing moves needed to complete Γ are feasible, we prove that two properties are invariant after every clearing move of S following s_c and corresponding clearing move of Γ, if any. For any move m of S that follows s_r, we call the sequence of moves in S through m an *initial portion of* S *beyond* s_r. We define a *corresponding initial portion of* Γ, which consists of all moves of Γ that correspond to moves in the initial portion of S.

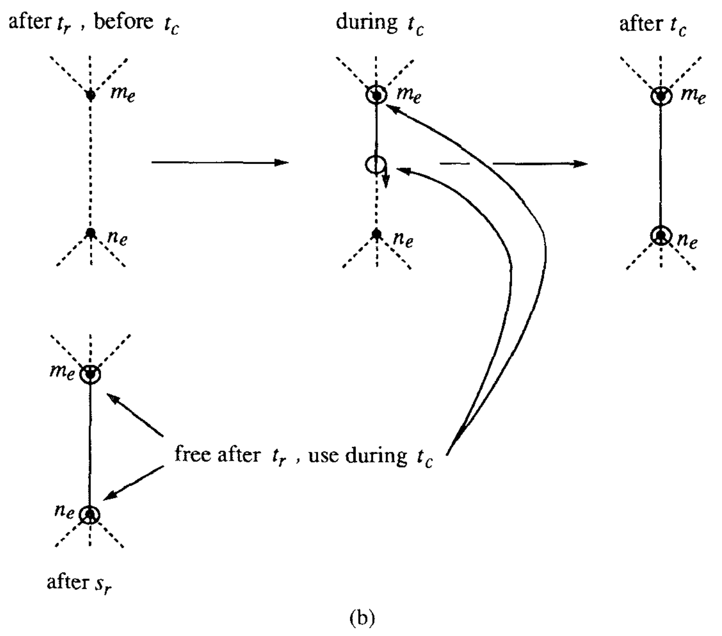
Claim. For any initial portion of S beyond s_r and corresponding initial portion of Γ, the following conditions hold:

(i) The set of edges that are clear at the end of the initial portion of S is a subset of the set of edges that are clear at the end of the corresponding portion of Γ;
(ii) If n is a node that is not guarded after the initial portion of S and is guarded after the corresponding portion of Γ, then n is an element of R.

We prove the claim by induction on the number of moves following s_r in the initial portion of S. The basis is the initial portion of S through s_r and Γ through s_c. Conditions (i) and (ii) hold since these sequences of moves only differ in that the portion of S ends with extra recontamination move s_r.

Consider a move m in S after s_r. By the induction hypothesis, (i) and (ii) hold just before m in S and its corresponding move, if any, in Γ. Suppose m clears an edge that is already clear for Γ. Then there is no corresponding move in Γ and the same portion of Γ that corresponded to the initial portion of S up to but not including m corresponds to the initial portion of S through m. Condition (i) holds after m since m clears an edge that is already clear under the corresponding portion of Γ. Suppose some node n becomes unguarded



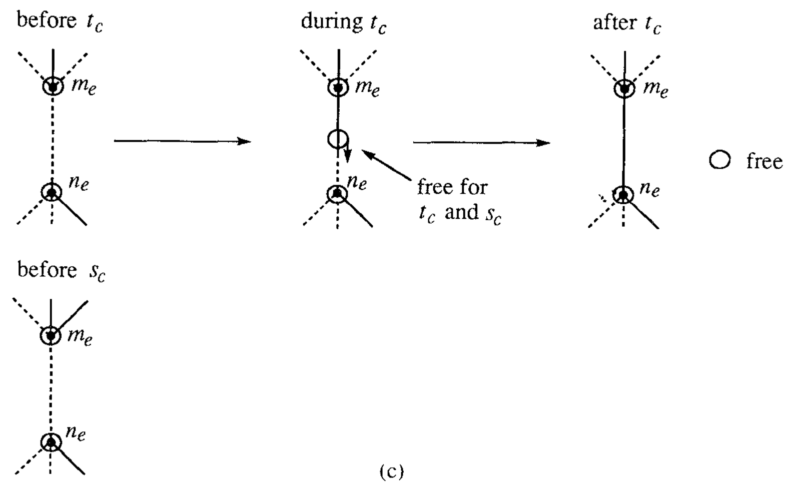


after t_r , before t_c
during t_c
after t_c
m_e
n_e
free after t_r , use during t_c
after s_r
(b)

FIG. 4. Examples for the new clearing moving t_c. (a) No free searches needed. (b) Two free searchers needed. (c) Both s_c and t_c require one free searcher. (d) t_c requires one free searcher; s_c does not.

during m. This can only happen because the node becomes clear. Since condition (i) holds, the node must also be clear after the corresponding portion of Γ, and therefore unguarded. Thus, no new nodes satisfy the hypothesis of condition (ii) after m.

Suppose m clears an edge, e, which is not already clear for Γ. Then, a corresponding clearing move, m', in Γ is required. We show below that this corresponding move is feasible, using the conditions (i) and (ii) for the initial portions before m and m'. Given that m' is feasible, condition (i) holds after m and m', since the same edge is cleared by both moves. Only the endpoints of e can become guarded or unguarded during m and m'. If an endpoint of e becomes unguarded during m, it is because it becomes clear, in which case it will be clear after m' also. An endpoint of e can become guarded,



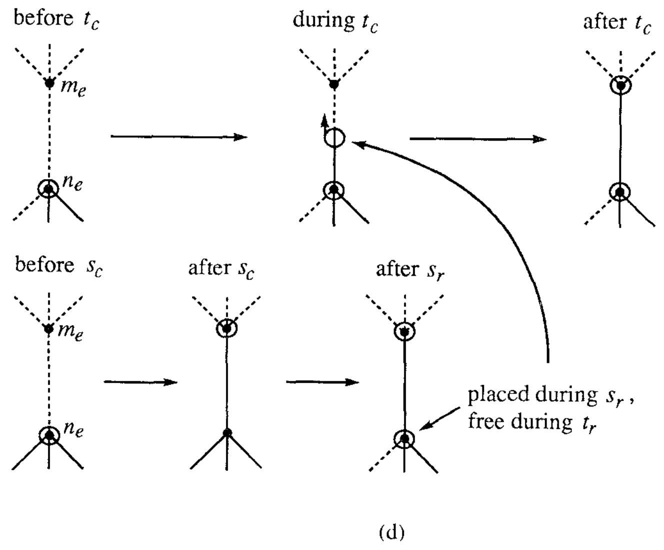


FIG. 4. (Continued).

so partially clear, after m'. Since e is cleared by m, the endpoint of e must be partially clear, so guarded, after m also. Therefore, no endpoint of e becomes guarded after m' and unguarded after m, and condition (ii) holds after m and m'.

To prove that m' is feasible, we first show that condition (ii) implies that the number of free searchers before m in S is no larger than the number of free searchers before m' in Γ. Consider any minimal path in $G - (N_{\mathrm{gr}} - R)$ from a node in R to a node in P. Since all nodes in P become partially clear during s_{r}, such a path must exist. We are interested in the state of nodes and edges along this path at various times during S and Γ (see Figure 5). Immediately after s_{c} (i.e., before s_{r}), all nodes and edges along this path are clear except for

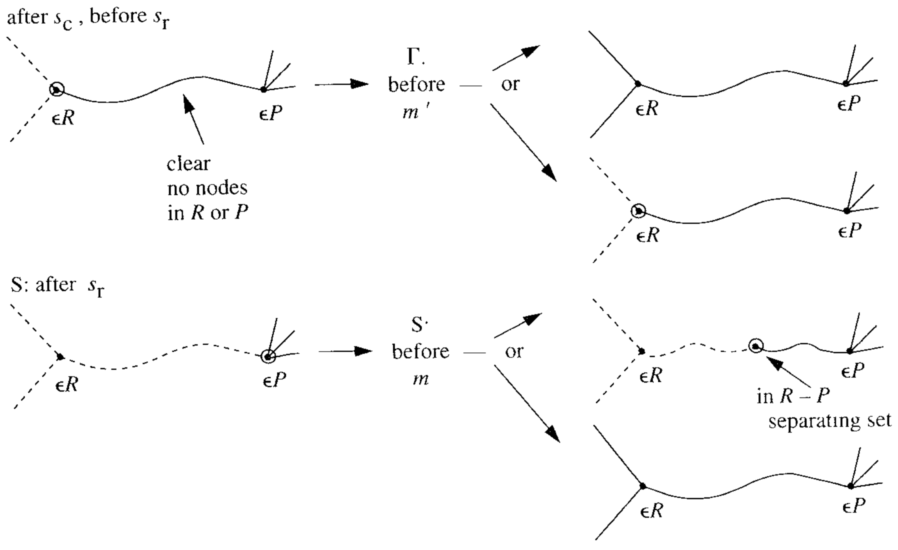


FIG. 5. States of a path from R to P.

the node in R, which is guarded. Therefore, at any point after s_c in Γ, the node in R is the only node that can be guarded along this path, and all edges on the path are clear. After s_r, all edges and nodes along the path are contaminated except for the element of P that is guarded. Consider the state of this path just before m. Either it is completely clear and contains no searchers, in which case (by condition (i)) the path is in the same state before m', or it contains at least one guarded node. If such a guarded node is not the element of R, then it is not guarded before m'; if the element of R is guarded, it may or may not be guarded before m'.

In any of the cases above, the following is true of the states of the path before m and m': either the states of the element of R agree, or the path contains a node that is guarded before m but not guarded before m'. Therefore, the set of all nodes of R whose states agree before m and m' in union with the set of all nodes that are guarded before m and not guarded before m' must form a separating set of R and P in $G - (N_{gr} - R)$, that is, an unguarded separating set of R and P. Using our hypothesis that no unguarded separating set is smaller than $|R|$, we have:

$$|\text{nodes in } R \text{ whose states agree before } m \text{ and } m'| + |\text{nodes that are guarded before } m \text{ but not before } m'| \geq |R|.$$

Hence, by subtraction:

$$|\text{nodes that are guarded before } m \text{ but not before } m'| \geq |\text{nodes in } R \text{ whose state disagree before } m \text{ and } m'|.$$

Using (ii), we observe that all nodes that are guarded before m' but not before m are in R so that:

$$\begin{aligned}&|\text{nodes that are guarded before } m \text{ but not before } m'|\\ &\quad \geq |\text{nodes that are guarded before } m' \text{ but not before } m|\\ &\qquad + |\text{nodes in } R \text{ that are guarded before } m \text{ but not before } m'|.\end{aligned}$$

Adding |nodes that are guarded before m and m'| gives:

$$\begin{aligned}&|\text{nodes that are guarded before } m|\\ &\quad \geq |\text{nodes that are guarded before } m'|\\ &\qquad + |\text{nodes in } R \text{ that are guarded before } m \text{ but not before } m'|\end{aligned}$$

and

$$\begin{aligned}&|\text{free searchers before } m| \leq |\text{free searchers before } m'|\\ &\quad - |\text{nodes in } R \text{ which are guarded before } m \text{ but not before } m'|\end{aligned}$$

$$|\text{free searchers before } m| \leq |\text{free searchers before } m'|.$$

To complete the proof of this lemma, we show that m' is feasible. Move m' is defined using three cases: If before m' one endpoint of e is partially clear and e is the only contaminated edge incident on this endpoint, then the searcher on this endpoint is moved to clear e and no free searchers are required. If both endpoints of e are of degree at least two and are contaminated before m', then m' requires two free searchers. Both searchers are put on one endpoint, and one is moved to the other endpoint to clear e. If neither of the preceding cases occurs, m' requires one free searcher. This free searcher is put on a degree-one endpoint or a guarded endpoint and moved along e.

If m' requires no more free searcher than m, then, since we have shown above that these free searchers are available, m' is feasible. Suppose, on the contrary, that m' does require more free searchers than m. Then either m' requires two free searchers or m requires no free searchers. If m' requires two free searchers, then by condition (i), both endpoints of e are also contaminated before m, and m requires two free searchers. If m requires no free searchers, then before m one endpoint of e is partially clear, and e is the only contaminated edge incident on this endpoint. By condition (i), this endpoint is in the same state before m', and m' requires no free searchers. Thus, assuming m' requires more free searchers than m leads to a contradiction. We conclude that m' is feasible. □

Our final technical lemma shows how we use Lemmas 4–7 to prove Theorem 1. It formalizes the idea of either postponing the clearing of edges or eliminating recontamination moves. For any search strategy, S, combine any sequence of recontamination moves not separated by a clearing move into one recontamination move using Lemma 4. Call these *combined recontamination moves*. Note that the number of edges recontaminated between any two clearing moves is unchanged. Then, for any search strategy S, define ordered pair $P(S) = (CM(S), ER(S))$, where $CM(S)$ is the number of clearing moves before the last combined (nonempty) recontamination move and $ER(S)$ is the number of edges recontaminated in the last combined recontamination move. If S has no recontamination moves, then $P(S) = (0,0)$. Order the values of $P(S)$ lexicographically, that is, for strategies S_1 and S_2, $P(S_1) < P(S_2)$ if and only if $CM(S_1) < CM(S_2)$ or $CM(S_1) = CM(S_2)$ and $ER(S_1) < ER(S_2)$.

LEMMA 8. *Given a graph and a search strategy S_1 for it using k searchers in which an edge is recontaminated, there is another search strategy S_2 for the graph using k searchers for which $P(S_2) < P(S_1)$.*

PROOF. Consider the last combined (nonempty) recontamination move. Assume it is irredundant, and let R and P be the sets for steps (1) and (2). If the number of free searchers after this move is at least as large as the number before it, then replace the preceding clearing move and the recontamination move with a recontamination move and following clearing move according to Lemma 6. If the new recontamination move is not empty, the number of clearing moves before the last combined recontamination move has decreased by one, regardless of whether the new clearing move is empty; if the new recontamination move is empty (no edges are recontaminated), the number of clearing moves before the last combined recontamination move has decreased by at least one.

If the number of free searchers is less after the recontamination move ($|R| < |P|$), there are two cases depending on whether or not there is an unguarded separating set of P and R containing less nodes than R. If there is, let C be a minimal such separating set. Decompose the recontamination move into two according to Lemma 5. Recall that the first removes from $R - C$ and places on $C - R$; the second removes from $C - P$ and places on $P - C$. Since $|C| < |R|$ and $|R| < |P|$, $R - C$ and $C - P$ are not empty, and some edge is recontaminated during each move. Since $|R - C| = |R| - |R \cap C| > |C| - |R \cap C| = |C - R|$, there are at least as many free searchers after the first move as before it. Use Lemma 6 to replace the first recontamination move and preceding clearing move with a recontamination move followed by a clearing move. If the new clearing move is empty, the two recontamination moves are recombined, but one less clearing move occurs before them. If the clearing move is not empty, the second recontamination move of the decomposition is the new last combined recontamination move. At least one less edge is recontaminated during it versus during the original last move, while the number of clearing moves before it is unchanged. In any of the cases, $P(S_2)$ has decreased.

If the number of free searchers is less after the last recontamination move, and there is no unguarded separating set C of R and P with $|C| < |R|$, then the last recontamination move can be eliminated according to Lemma 7. The clearing moves originally before this move but after a previous recontamination move (if any) are now after the last recontamination move. Therefore, the number of clearing moves before the last recontamination move is decreased by at least one. □

We now have all the technical lemmas needed to prove Theorem 1:

THEOREM 1. *If there is a search strategy for a graph under the new rules using k searchers, then there is a search strategy for the graph without any recontamination moves and still using k searchers.*

PROOF. Among all search strategies for a graph that use k searchers, consider one with lowest $P(S)$. By Lemma 8, $P(S) = (0, 0)$ for this strategy and the strategy has no recontamination moves. □

4. *Conclusion*

We have shown that a graph can always be searched using a minimum number of searchers without clearing any edge more than once. To do this, we have defined several transformations of move sequences that use recontamination. These transformations may be useful for other analyses of graph searching. However, it is usually simpler to study graph-searching problems by considering only search strategies without recontamination. The major contribution of this paper is that it allows one to do this without sacrificing minimality of searchers.

ADDENDUM: Since the submission and circulation of this paper, Bienstock and Seymour have developed an alternative, shorter proof of the result; their proof is presented in [1].

ACKNOWLEDGMENTS. I would like to thank Mihalis Yannakakis for helpful discussions in the development of this proof. I would also like to thank Michael Garey, David Johnson, and the referees for suggestions improving the explication.

REFERENCES

1. BIENSTOCK, D., AND SEYMOUR, P. Monotonicity in graph searching. *J. Algorithms 12* (1991), 239–245.
2. BREISCH, R. An intuitive approach to speleotopology. *Southwestern Cavers* (A publication of the Southwestern Region of the National Speleological Society) *VI*, 5 (Dec. 1967), 72–78.
3. ELLIS, J. A., SUDBOROUGH, I. H., AND TURNER, J. S. Graph separation and search number. In *Proceedings of the Allerton Conference on Communication, Control and Computing*. Coordinated Sci. Lab., Univ. Illinois, Urbana-Champaign, Ill., 1983, pp. 224–233.
4. GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, Calif., 1979.
5. KIROUSIS, L., AND PAPADIMITRIOU, C. Searching and pebbling. *Theoret. Comput. Sci. 47* (1986), 205–218.
6. MAKEDON, F. Layout problems and their complexity. Ph.D. dissertation. Dept. of Electrical Engineering and Computer Science, Northwestern Univ., Chicago, Ill., 1982.
7. MAKEDON, F., PAPADIMITRIOU, C., AND SUDBOROUGH, I. H. Topological bandwidth. *SIAM J. Alg. Disc. Meth. 6* (1985), 418–444.
8. MAKEDON, F., AND SUDBOROUGH, I. H. On minimizing width in linear layouts. *Disc. Appl. Math. 23* (1989), 243–265.
9. MEGIDDO, N., HAKIMI, S. L., GAREY, M. R., JOHNSON, D. S., AND PAPADIMITRIOU, C. H. The complexity of searching a graph, *J. ACM 35*, 1 (Jan. 1988), 18–44.
10. PARSONS, T. D. Pursuit-evasion in a graph. In *Theory and Applications of Graphs*. Y. Alavi and D. R. Lick, eds. Lecture Notes in Mathematics. Springer-Verlag, New York, 1976, pp. 426–441.
11. PARSONS, T. D. The search number of a connected graph. In *Proceedings of the 9th Southeastern Conference on Combinatorics, Graph Theory and Computing*. Utilitas Mathematica Publishing, Winnipeg, Canada, 1978, pp. 549–554.

RECEIVED JANUARY 1985; REVISED SEPTEMBER 1991; ACCEPTED SEPTEMBER 1991