Theoretical Computer Science 495 (2013) 37–49

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On minimum cost edge searching

Dariusz Dereniowski [a,*,1], Danny Dyer [b,2]

[a] Department of Algorithms and System Modeling, Gdańsk University of Technology, Gdańsk, Poland
[b] Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's, Canada

ARTICLE INFO

Article history:
Received 30 July 2010
Received in revised form 8 June 2013
Accepted 18 June 2013
Communicated by G. Ausiello

Keywords:
Approximation algorithm
Graph searching
Minimum cost
Monotonicity
Search strategy

ABSTRACT

We consider the problem of finding edge search strategies of minimum cost. The cost of a search strategy is the sum of searchers used in the clearing steps of the search. One of the natural questions is whether it is possible to find a search strategy that minimizes both the cost and the number of searchers used to clear a given graph G. We call such a strategy *ideal*. We prove, by an example, that ideal search strategies do not exist in general. On the other hand, we provide a formula for the cost of clearing complete graphs. From our construction it follows that an ideal search strategy of a complete graph does exist and can be calculated efficiently. For general graphs G we give a polynomial-time $O(\log n)$-approximation algorithm for finding minimum cost search strategies. We also prove that recontamination does not help to obtain minimum cost edge search strategies.

1. Introduction

Edge searching was first introduced by Parsons in [9]. This model involves moving a number of searchers through a graph to find a fast, invisible intruder. Searching has shown itself to be very versatile however, and has shown up in several unlikely places. Edge searching is closely related to the graph parameter pathwidth, and hence to the broad theory of graph minors developed by Robertson and Seymour [11]. Edge searching has also been linked to applications in memory allocation (through pebbling [7]), to VLSI theory [3], and, at its most rudimentary, to finding children lost in a cave [2].

In Parsons' original paper, the graphs considered were embedded in $\mathbf{R}^3$ and the movement of the searchers and intruders were described by continuous functions. However, this is not the form in which this model is normally considered; instead, we describe the movement of the searchers by discrete actions. For formal definitions see Section 2.

The classical measure for the quality of a search strategy is the number of searchers it uses. This naturally follows from the need to minimize the size of a team in applications, as well as from the connections between search numbers and width-like graph parameters, including the cutwidth, bandwidth, pathwidth and treewidth. However, other optimality criteria for search strategies are also interesting. One such criterion is the maximum occupation time introduced in [5], which is related to computing elimination trees of a graph and to the graph parameter *treespan*. Another is the cost of node searching [4]. As proved in [4], the cost is related to the *interval completion* problem, that is, finding the minimum number of edges required to add to a given graph G to obtain an interval supergraph of G, or the *profile* of G. The aim of this paper is to adopt the concept of the cost to the edge searching problem.

This paper is organized as follows. In Section 2 we recall the edge and node searching problems. In Section 3 we investigate the problem of monotonicity, which is a basic question for any model of graph searching: we prove that

* Corresponding author.
E-mail addresses: deren@eti.pg.gda.pl (D. Dereniowski), dyer@mun.ca (D. Dyer).
[1] Partially supported by MNiSW grant N N206 379337 (2009–2011).
[2] Supported by NSERC.

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2013.06.009

recontamination does not help while constructing minimum cost search strategies. Then, in Section 4, we transform the problem of finding edge search strategies of minimum cost to the problem of minimizing the cost of node-searching, which is related to finding the profile of a graph [4]. The best known approximation algorithm for the two latter problems has the ratio $O(\log n)$ and, due to our transformation, we obtain an $O(\log n)$-approximation algorithm for finding minimum cost edge search strategies. Section 5 provides a formula for the cost of edge searching complete graphs. The proof is constructive, i.e., we also give an algorithm that finds an edge search strategy of minimum cost (the strategy also uses the minimum number of searchers). Section 6 shows that minimizing the number of searchers and minimizing the cost are 'orthogonal' in the sense that there exist graphs G such that each search strategy of minimum cost does not use the minimum number of searchers.

2. Preliminaries

In this section we formally define the edge and node searching problems, together with the two optimization criteria considered in this work. In both of these graph searching models we assume that the searchers have complete knowledge of the graph and each other's locations. The intruder is assumed to be invisible to the searchers, and may stop on vertices or edges. The intruder may move from its current position along any path that is not occupied by a searcher, at any time. Also, the intruder is omniscient, i.e., it has the complete knowledge about the locations and the future moves of the searchers, and hence it will avoid being captured as long as possible.

For any graph G, $V(G)$ and $E(G)$ denote its vertex and edge sets, respectively. Given any graph $G = (V(G), E(G))$ and a vertex $v \in V(G)$, E_v denotes the set of edges incident to v.

We say that $\mathcal{S}$ is an *edge search strategy* (or *search strategy* for short) of a graph G if $\mathcal{S}$ is a sequence of moves of the searchers that guarantee capture of the intruder, i.e., the intruder and some searcher will share their locations at some point as a result of the execution of $\mathcal{S}$, regardless of the actions of the intruder. There are three possible moves in $\mathcal{S}$:

(i) a single searcher is placed on an arbitrary vertex of G,
(ii) a single searcher is removed from an arbitrary vertex of G,
(iii) a single searcher σ present on a vertex u slides along an edge $uv \in E(G)$, ending at v (in such case we also say that σ *slides from u to v*). We call such a move a *clearing move*.

(By combining the first and second moves, a searcher may "jump" from one vertex to a non-adjacent vertex.) By the definition, the searchers only stop on vertices. The symbol $\mathtt{s}(\mathcal{S})$ denotes the number of searchers that $\mathcal{S}$ uses. The (*edge*) *search number* of a graph G is

$$\mathtt{s}(G) = \min\{\mathtt{s}(\mathcal{S}) \colon \mathcal{S} \text{ is a search strategy of } G\}.$$

An edge uv is *cleared*, or guaranteed to be free of the intruder, in one of two ways: two searchers σ_1 and σ_2 are located on u, and σ_1 slides along uv; or, a single searcher σ_1 is located on u, and all edges in E_u other than uv are already cleared, and σ_1 slides along uv. Initially, all edges are *contaminated*; they may hold the intruder. At the end of a search strategy, all edges are clear, and thus the intruder's capture has been guaranteed.

A cleared edge uv in a graph G reverts to being contaminated if there is ever a path from u (or correspondingly, v) to a vertex x of a contaminated edge xy such that the path contains no searchers. Then we say the edge has been *recontaminated*. A search strategy is *monotone* if no edge ever becomes recontaminated. If we insist on monotonicity in our search strategy, we may similarly define the *monotone (edge) search number* $\mathtt{ms}(G)$ of a graph G as the minimum number of searchers needed for a monotone search strategy of G. It is well known that $\mathtt{ms}(G) = \mathtt{s}(G)$ [1,8].

Given an edge search strategy, we use the symbol $|\mathcal{S}|$ to denote the number of moves in $\mathcal{S}$. Then, $\mathcal{S}_i$ denotes its i-th move, while $|\mathcal{S}_i|$ is the number of searchers used in this move, $i = 1, \ldots, |\mathcal{S}|$. For each $i = 1, \ldots, |\mathcal{S}|$ define $a(\mathcal{S}_i)$ to be $|\mathcal{S}_i|$ if $\mathcal{S}_i$ is a clearing move, and to be 0 otherwise, i.e., when $\mathcal{S}_i$ places or removes a searcher. The *cost* of an edge search strategy $\mathcal{S}$ is

$$\mathrm{cost}(\mathcal{S}) = \sum_{i=1}^{|\mathcal{S}|} a(\mathcal{S}_i). \tag{1}$$

Then, the *edge search cost* of a graph G is

$$\mathrm{cost}(G) = \min\{\mathrm{cost}(\mathcal{S}) \colon \mathcal{S} \text{ is a search strategy of } G\}. \tag{2}$$

We may also define the *monotone edge search cost* of edge searching a graph G as

$$\mathrm{cost}_{\mathrm{m}}(G) = \min\{\mathrm{cost}(\mathcal{S}) \colon \mathcal{S} \text{ is a monotone search strategy of } G\}. \tag{3}$$

Some graphs G may be such that there exists a search strategy $\mathcal{S}$ that simultaneously minimizes $\mathrm{cost}(G)$ and $\mathtt{s}(G)$, i.e., $\mathrm{cost}(\mathcal{S}) = \mathrm{cost}(G)$ and $\mathtt{s}(\mathcal{S}) = \mathtt{s}(G)$. Such a strategy $\mathcal{S}$ is called *ideal*. We will exhibit an ideal search strategy for the

complete graph. However, in general, graphs do not have ideal search strategies. We construct an infinite family of graphs for which no search strategy using $\mathrm{s}(G)$ searchers will simultaneous minimize the cost of cleaning G.

Now we recall the node searching problem and its cost measure as defined in [4]. We say that $\mathcal{N}$ is a *node search strategy* if it is a sequence of moves of the searchers that guarantees the capture of the intruder, which occurs when a searcher and the intruder are on the same vertex of G, or the intruder is on an edge whose two endpoints are occupied by searchers. In a node search strategy each move is the following sequence of actions:

(i) first, a single searcher is placed on a vertex $v \in V(G)$ that may contain the intruder;
(ii) second, any subset of searchers is removed from the vertices of G.

Then $\mathrm{s}(\mathcal{N})$ is the number of searchers used by $\mathcal{N}$. The *node search number* of a graph G is

$$\mathrm{ns}(G) = \min\{\mathrm{s}(\mathcal{N}) \colon \mathcal{N} \text{ is a node search strategy of } G\}.$$

As for edge search, we say that a node search strategy is *monotone* if the intruder is unable to reach any edge that has been previously cleared.

For a node search strategy $\mathcal{N}$, $|\mathcal{N}|$ denotes its length, $\mathcal{N}_i$ is its i-th move, and $|\mathcal{N}_i|$ denotes the number of searchers in use at the end of the move $\mathcal{N}_i$ (i.e., after performing both actions (i) and (ii)), $i = 1, \ldots, |\mathcal{N}|$. Following the notation in [4], we use the symbol $\gamma(\mathcal{N})$ to denote the *cost* of a node search strategy $\mathcal{N}$ clearing a graph G, defined as $\gamma(\mathcal{N}) = \sum_{1 \leqslant i \leqslant |\mathcal{N}|} |\mathcal{N}_i|$. We define the *node search cost* of a graph G as

$$\gamma(G) = \min\{\gamma(\mathcal{N}) \colon \mathcal{N} \text{ is a node search of } G\}.$$

We finish this section by introducing the three 'states' of a vertex during the execution of an edge or a node search strategy. A vertex of a graph is *contaminated* if no searcher is located at v and some edge in E_v is contaminated (in which case all edges incident to the vertex are contaminated). We say that a vertex v is *guarded* if a searcher is located at v. If all edges adjacent to a vertex v have been cleared, then v is *clear*. Note that once a vertex is clear, no searcher should guard v in an optimal monotone minimum cost (node or edge) search strategy. In other words, in the case of edge search, if a searcher σ slides from v to u and uv is the only contaminated edge in E_v at the beginning of this move, then no additional searcher guards v while σ performs the clearing of uv. Moreover, in each monotone search strategy each vertex $v \in V(G)$ is initially contaminated, then becomes guarded and finally it becomes clear once all the edges in E_v are cleared.

3. Monotonicity

In this section we prove that the edge search cost equals the monotone edge search cost for any graph G, $\mathrm{cost}(G) = \mathrm{cost}_{\mathrm{m}}(G)$.

Before we describe the details of our proof, we sketch our method. The 'standard' method of proving monotonicity for graph searching models is the concept of crusades introduced by Bienstock and Seymour in [1]. This method turned out to be successful in proving that recontamination does not help for minimum cost node searching [4]. We prove the monotonicity for minimum cost edge searching by a 'reduction' to the node search problem. In particular, we define a graph G_p for a given graph G by replacing each edge by a path on p edges. If the integer p is sufficiently large, then node and edge searching programs 'behave' in a very similar way for G_p, that is, the difference between node and edge costs is very small compared to the additional cost introduced by a possible recontamination. Since recontamination does not help for minimum cost node searching of G_p, we will obtain that recontamination does not help for minimum cost edge searching in the class of graphs G_p. Moreover, we prove that $\mathrm{cost}(G_p) = p \cdot \mathrm{cost}(G)$ and finally we obtain our main result by a contradiction: $\mathrm{cost}(G) < \mathrm{cost}_{\mathrm{m}}(G)$ would imply that recontamination helps for minimum cost (node and edge) searching of G_p as well.

Given a simple graph G, construct a graph G_p by replacing each edge uv of G by a path on p edges with endpoints u and v, or alternatively, by subdividing the edge uv $p-1$ times. Note that in particular $G_1 = G$. Informally, Lemmas 1 and 2 allow us to assume that, in a node or edge search strategy for graphs G_p, if a search strategy starts clearing a path of G_p corresponding to an edge of G, then the strategy continues by clearing the path completely. To give the formal statements we introduce some more notation. We say that the edges of a path P with vertices $v_1, \ldots, v_n$ and edges $v_1v_2, v_2v_3, \ldots, v_{n-1}v_n$ are *cleared consecutively* in an edge search strategy $\mathcal{S}$ if, for some $j \in \{1, \ldots, |\mathcal{S}| - n + 2\}$, S_j clears v_1v_2, S_{j+1} clears v_2v_3, and so on, until S_{j+n-2} clears $v_{n-1}v_n$. Analogously, the vertices of P are *guarded consecutively* in a node search strategy $\mathcal{N}$ if for some $j \in \{1, \ldots, |\mathcal{N}| - n + 1\}$, N_j guards vertex v_1, N_{j+1} guards v_2, and so on, until N_{j+n-1} guards v_n.

Lemma 1. *If G is a graph, $p \geqslant 1$ is an integer, and $\mathcal{N}$ is a monotone node search strategy of G_p, then there exists a monotone node search strategy $\mathcal{N}'$ of G_p such that $\gamma(\mathcal{N}') \leqslant \gamma(\mathcal{N})$, $\mathrm{s}(\mathcal{N}') = \mathrm{s}(\mathcal{N})$ and the internal vertices of each path P corresponding to an edge of G are guarded consecutively by $\mathcal{N}'$.*

Proof. Let $\widetilde{\mathcal{N}}$ be a monotone node search strategy of G_p whose sum $\sum_{v\in V(G)} t(v)$ is minimum and such that $\gamma(\widetilde{\mathcal{N}}) \leqslant \gamma(\mathcal{N})$, $\mathrm{s}(\widetilde{\mathcal{N}}) \leqslant \mathrm{s}(\mathcal{N})$, where $\widetilde{\mathcal{N}}_{t(v)}$ is the move of placing a searcher on v, $v \in V(G)$. If $\widetilde{\mathcal{N}} \neq \mathcal{N}$, then replace $\mathcal{N}$ by $\widetilde{\mathcal{N}}$.

Let the path P connecting two vertices u, v of G_p and corresponding to an edge $uv \in E(G)$ be selected arbitrarily, and let P' be its subpath on the internal vertices of P. (Note that $V(P') = V(P) \setminus \{u, v\}$.) Since the lemma trivially holds when $p \leqslant 2$, let $p \geqslant 3$ in the following. Let u' and v' be the endpoints of P' adjacent to u and v, respectively. Further, without loss of generality, assume that u is guarded before v in $\mathcal{N}$. Note that any node search strategy of G_p consists of n moves, where n is the number of vertices of G_p.

We first prove that there exists a node search strategy $\mathcal{N}''$ of G_p such that for each $i = 1, \ldots, n$ it holds:

(i) at the end of $\mathcal{N}_i''$ either no vertex of P' is guarded, or the cleared edges and guarded vertices of P' form a connected subpath that contains u'. (Essentially, the path P' of internal vertices of P is cleared from one side to the other in $\mathcal{N}''$, and not by starting in the middle.)
(ii) $B(\mathcal{N}_i'') \setminus V(P) = B(\mathcal{N}_i) \setminus V(P)$, where $B(\mathcal{N}_i)$ and $B(\mathcal{N}_i'')$ are the sets of vertices guarded at the end of the moves $\mathcal{N}_i$ and $\mathcal{N}_i''$, respectively.
(iii) $|\mathcal{N}_i''| \leqslant |\mathcal{N}_i|$.

Observe that (iii) implies $\gamma(\mathcal{N}'') \leqslant \gamma(\mathcal{N})$.

We construct $\mathcal{N}''$ by defining each of its moves. Suppose that $\mathcal{N}_1'', \ldots, \mathcal{N}_{i-1}''$ have been constructed, where $i \in \{1, \ldots, n\}$, and we define $\mathcal{N}_i''$. If $\mathcal{N}_i$ places a searcher on a vertex not in $V(P')$ then so does $\mathcal{N}_i''$. One can check that $\mathcal{N}_i''$ satisfies conditions (i)–(iii).

Suppose now that $\mathcal{N}_i$ places a searcher on a vertex x of P'. If none of $\mathcal{N}_1, \ldots, \mathcal{N}_{i-1}$ placed a searcher on a vertex of P', then $\mathcal{N}_i''$ places a searcher on u'. Clearly, (i)–(iii) hold because x needs to be guarded at the end of $\mathcal{N}_i$.

Hence suppose that at least one of $\mathcal{N}_1, \ldots, \mathcal{N}_{i-1}$ placed a searcher on a vertex of P'. Let P'' be the subpath of P' consisting of all vertices guarded by $\mathcal{N}_1'', \ldots, \mathcal{N}_{i-1}''$. By construction, u' is an endpoint of P'', and let v'' be the vertex in $V(P') \setminus V(P'')$ adjacent to an endpoint of P''. We define $\mathcal{N}_i''$ as a move that places a searcher on v''. Note that (i) holds, and (ii) follows from (ii) for $i - 1$. It remains to prove (iii). Note that, by construction,

$$u \in B\big(\mathcal{N}_i''\big) \quad \Rightarrow \quad u \in B(\mathcal{N}_i). \tag{4}$$

We consider two cases: $v \in B(\mathcal{N}_i)$ and $v \notin B(\mathcal{N}_i)$. If $v \in B(\mathcal{N}_i)$, then $v \in B(\mathcal{N}_i'')$ and hence

$$\big|B\big(\mathcal{N}_i''\big) \cap \big(V(P) \setminus \{u\}\big)\big| = 2 \leqslant \big|B(\mathcal{N}_i) \cap \big(V(P) \setminus \{u\}\big)\big|$$

and (iii) follows from (ii) and (4). If $v \notin B(\mathcal{N}_i)$, then suppose first that v is not clear at the end of $\mathcal{N}_i$. If u is guarded or clear at the end of $\mathcal{N}_i$, then, by (4), $B(\mathcal{N}_i'') \cap (V(P) \setminus \{u\}) = \{v''\}$ and, since $B(\mathcal{N}_i) \cap (V(P) \setminus \{u\}) \neq \emptyset$, (iii) follows from (ii) and (4). If u is not guarded and not clear at the end of $\mathcal{N}_i$, then $B(\mathcal{N}_i'') \cap V(P) = \{u', v''\}$ and hence $|B(\mathcal{N}_i'') \cap V(P)| > 1$ and (ii) also imply (iii). It remains to consider the case when $v \notin B(\mathcal{N}_i)$ and v is clear at the end of $\mathcal{N}_i$. Let $y \in B(\mathcal{N}_i)$ be the endpoint of the maximal subpath of P' consisting of cleared edges and guarded vertices at the end of $\mathcal{N}_i$ and having v' as an endpoint. Note that $B(\mathcal{N}_i'') \cap (V(P) \setminus \{u\}) \subseteq \{v, v''\}$ because u is guarded or clear at the end of $\mathcal{N}_i''$. If $|B(\mathcal{N}_i) \cap (V(P) \setminus \{u\})| > 1$, then (iii) follows from (ii) and (4). Thus, suppose that $|B(\mathcal{N}_i) \cap (V(P) \setminus \{u\})| = 1$ (or equivalently, $B(\mathcal{N}_i) \cap (V(P) \setminus \{u\}) = \{y\}$) and $B(\mathcal{N}_i'') \cap (V(P) \setminus \{u\}) = \{v, v''\}$. Since u is guarded prior to v in $\mathcal{N}$, we obtain that some subpath of P' with and endpoint v' has been cleared by $\mathcal{N}$ before a searcher has been placed on v in $\mathcal{N}$, which contradicts the choice of $\mathcal{N}$ (this subpath could be selected in such a way that v, not v', is its endpoint).

Now we prove the lemma. Let $\mathcal{N}_i''$, $i \in \{1, \ldots, n\}$, be a move placing a searcher on an internal vertex x of P, which results in clearing an edge of P and guarding at least one vertex of P at the end of $\mathcal{N}_i''$. Moreover, select i so that the number of searchers used for guarding at the end of $\mathcal{N}_i''$ is minimum over all such moves in $\mathcal{N}''$. Note that i is well defined because $p \geqslant 3$. Let $\mathcal{N}_j''$ and $\mathcal{N}_k''$ be the moves that place a searcher on a vertex of P' and, as a result, no edge of P becomes cleared and all edges become cleared, respectively. (Note that $\mathcal{N}_j''$ and/or $\mathcal{N}_k''$ may not exist, as the first and last clearing moves may be caused by placing a searcher on the endpoints of P and not in P'; in such case we say that they are undefined.) Let $q(\mathcal{N}_i'') = 0$, $q(\mathcal{N}_j'') = 1$ and let $q(\mathcal{N}_k'') = 2$. Let $l \in \{i, j, k\}$ be selected in such a way that $\mathcal{N}_l''$ is not undefined and $|\mathcal{N}_l''| + q(\mathcal{N}_l'') = \min\{|\mathcal{N}_t''| + q(\mathcal{N}_t'')\colon\ t = i, j, k$ and $\mathcal{N}_t''$ is not undefined$\}$. To obtain $\mathcal{N}'$, reorder the moves in $\mathcal{N}''$ so that all moves of clearing the vertices of P replace the move $\mathcal{N}_l''$ and the vertices of P are guarded consecutively. By the choice of l, the total cost of clearing and guarding the vertices of P is not greater in the new strategy than in the initial one. Hence, $\gamma(\mathcal{N}') \leqslant \gamma(\mathcal{N}'') \leqslant \gamma(\mathcal{N})$.

The above modification can be repeated independently for each path corresponding to an edge of G. This completes the proof. □

Note that Lemma 1 specifies a way of clearing the internal vertices of the paths in G_p that correspond to the edges of G, though a move that places a searcher on an endpoint of such path (i.e., on a vertex of G) does not have to be followed immediately by clearing the path since the endpoint may be shared by several such paths. A similar result holds for edge searching of G_p.

Lemma 2. *If G is a graph, $p \geqslant 1$, and $\mathcal{S}$ is an edge search strategy of G_p, then there exists an edge search strategy $\mathcal{S}'$ of G_p such that $\mathrm{cost}(\mathcal{S}') \leqslant \mathrm{cost}(\mathcal{S})$, $\mathrm{s}(\mathcal{S}') = \mathrm{s}(\mathcal{S})$ and, in $\mathcal{S}'$, when any edge in a path corresponding to an edge in G is cleared, the rest of the edges in that path are consecutively cleared, and when any edge in a path corresponding to an edge in G is recontaminated, all edges in that path are recontaminated.*

Lemma 3. *If G is a graph and $p \geqslant 1$ is an integer, then there exists a (monotone) edge search strategy $\mathcal{S}$ of G with $\mathrm{cost}(\mathcal{S}) = k$ if and only if there exists a (monotone) edge search strategy $\widetilde{\mathcal{S}}$ of G_p with $\mathrm{cost}(\widetilde{\mathcal{S}}) = pk$.*

Proof. Given an edge search strategy $\mathcal{S}$ that clears G, we construct an edge search strategy $\widetilde{\mathcal{S}}$ of G_p as follows. If $\mathcal{S}_i$ clears an edge uv by sliding σ from u to v, then add to $\widetilde{\mathcal{S}}$ the p moves that clear the path with p edges corresponding to uv by having σ slide from u to v through successive moves along the edges of P. Alternatively, if $\mathcal{S}$ places/removes a searcher σ at/from a vertex u, then $\widetilde{\mathcal{S}}$ places/removes σ at/from u, respectively. Clearly, $\mathrm{cost}(\widetilde{\mathcal{S}}) = p \cdot \mathrm{cost}(\mathcal{S})$. Moreover, if $\mathcal{S}$ is monotone, then $\widetilde{\mathcal{S}}$ is monotone as well.

Suppose now that a search strategy $\mathcal{S}'$ that clears G_p is given. By Lemma 2, there exists a search strategy $\widetilde{\mathcal{S}}$ with $\mathrm{s}(\widetilde{\mathcal{S}}) = \mathrm{s}(\mathcal{S}')$ and $\mathrm{cost}(\widetilde{\mathcal{S}}) \leqslant \mathrm{cost}(\mathcal{S}')$ and such that $\widetilde{\mathcal{S}}$ can be divided into sequences of moves, such that each such sequence is either a single move that places or removes a searcher on/from a vertex in $V(G)$, or it consists of clearing moves that slide a searcher from a vertex u to a vertex v (both in $V(G)$) along a path P in G_p. Construct $\mathcal{S}$ as follows; if $\widetilde{\mathcal{S}}_i$ is a move that places/removes a searcher σ at/from a vertex u, then add a move to $\mathcal{S}$ that places/removes σ at/from u. If $\widetilde{\mathcal{S}}_i$ is the beginning of a sequence of clearing moves from u to v along a path P, then add to $\mathcal{S}$ the single clearing move from u to v along the edge that corresponds to this path P. Then, each clearing move in $\mathcal{S}$ corresponds to a segment of p clearing moves in $\widetilde{\mathcal{S}}$, each of which uses the same set of searchers. Moreover, an edge of G gets recontaminated during $\mathcal{S}$ if and only if the corresponding path in G_p gets recontaminated in $\widetilde{\mathcal{S}}$. Thus, $p \cdot \mathrm{cost}(\mathcal{S}) = \mathrm{cost}(\widetilde{\mathcal{S}})$. □

We will prove in Lemma 5 that we can convert an edge search strategy of G_p into a node search strategy of G_p and the cost of the new strategy differs additively by a factor independent of p. In order to give this result for non-monotone search strategies we first give an upper bound on the number of sliding moves in a minimum cost edge search.

Lemma 4. *If G is a graph on n vertices, then each minimum cost edge search strategy of G contains at most n^3 clearing moves.*

Proof. First observe, that n^3 is an upper bound on the edge search cost of G. Indeed, one can construct a 'trivial' edge search strategy that first places a searcher on each vertex of G, and an additional $(n+1)$-st searcher clears all the edges while each vertex is guarded. The number of clearing moves is $m \leqslant n(n-1)/2$, which proves the bound.

Each clearing move of any minimum cost edge search strategy $\mathcal{S}$ contributes at least one unit to the overall cost of $\mathcal{S}$, because at least one searcher is used in the clearing move. Therefore, if there are more than n^3 clearing moves in $\mathcal{S}$, then $\mathrm{cost}(\mathcal{S}) > n^3$ – a contradiction. □

Lemma 5. *If G is a graph on n vertices, $p \geqslant 1$ is an integer, and $\mathcal{S}$ is a (monotone) edge search strategy of G_p, then there exists a (monotone) node search strategy $\mathcal{N}$ of G_p such that*

$$\gamma(\mathcal{N}) \leqslant \mathrm{cost}(\mathcal{S}) + 2n^3(n+1).$$

Proof. We begin by placing more structure on the edge search strategy $\mathcal{S}$. First, without loss of generality, we may assume that whenever $\mathcal{S}$ slides a searcher σ from u to v along an edge uv, the edge uv is cleared. (Otherwise, we could replace the sliding move by two moves: first, removing σ from u, then placing σ at v, which would give a lower cost search strategy.) Second, by Lemma 2, we may without loss of generality assume that the edges of each path in G_p that corresponds to an edge of G are cleared consecutively in $\mathcal{S}$.

Given such an edge search strategy $\mathcal{S}$ of G_p, we define a node search strategy $\mathcal{N}$ of G_p. Initially no vertex of G_p is occupied both in $\mathcal{S}$ and in $\mathcal{N}$. Suppose that the first $j-1$ moves of $\mathcal{N}$ have been obtained from the first $i-1$ moves of $\mathcal{S}$, for some $i, j \geqslant 1$. Now we define the moves of $\mathcal{N}$ that correspond to $\mathcal{S}_i$. If $\mathcal{S}_i$ places a searcher on a vertex, $\mathcal{N}$ does nothing. If $\mathcal{S}_i$ removes a searcher and this does not cause recontamination, then $\mathcal{N}$ does nothing. If $\mathcal{S}_i$ removes a searcher which results in recontamination, then in $\mathcal{N}$ we remove the searchers occupying all vertices v such that all of the edges that became recontaminated in $\mathcal{S}$ now become recontaminated in $\mathcal{N}$. If $\mathcal{S}_i$ slides a searcher σ along an edge from u to v, and both u and v are already occupied by searchers at the end of $\mathcal{N}_{j-1}$, then $\mathcal{N}$ does nothing (i.e., no moves are added to $\mathcal{N}$ while 'processing' $\mathcal{S}_i$; this edge has been already cleared in $\mathcal{N}$). On the other hand, if either u or v (or both) are not guarded at the end of $\mathcal{N}_{j-1}$, then in subsequent moves in $\mathcal{N}$ (i.e., in the move $\mathcal{N}_j$, or in the moves $\mathcal{N}_j, \mathcal{N}_{j+1}$ if none of u and v are guarded), place searchers on each of u and v that are unoccupied. (This may clear other edges, but will certainly clear uv.) Also, at the end of each move of $\mathcal{N}$ the searchers that are not necessary for guarding are removed, and therefore have no influence on the cost of the node search strategy.

Note that the above in particular implies that if a searcher reaches in $\mathcal{S}$ a vertex v for the first time by sliding the only searcher located at u along the edge $uv \in E(G_p)$, then a second searcher is placed on v in $\mathcal{N}$, and subsequently the single

searcher at u is immediately removed. It follows from this construction that if one move $\mathcal{N}_j$ in $\mathcal{N}$ corresponds to $\mathcal{S}_i$, then $|\mathcal{N}_j| \leqslant |\mathcal{S}_i|$. Therefore, clearing the subpaths induced by the internal vertices of the path of G_p corresponding to an edge of G gives the same cost both in $\mathcal{S}$ and in $\mathcal{N}$.

We now consider the cost of clearing those edges in G_p incident to the vertices of G, which are the only edges we have not yet considered. Note that the edges of paths of G_p that correspond to the edges of G are cleared consecutively by $\mathcal{S}$, and if a recontamination occurs, then all edges of such a path either remain cleared or become recontaminated. The number of times such a path in G_p is cleared, by Lemmas 3 and 4, is bounded by n^3. Moreover, to clear each edge of G_p incident to a vertex in $V(G)$, there exist at most two corresponding moves in $\mathcal{N}$, each using at most $n+1$ searchers. (There could be a searcher on each vertex in $V(G)$, as well as on one vertex not in $V(G)$.) Thus, each of these $2n^3$ moves uses at most $n+1$ searchers. □

Lemma 6. *If G is a graph with n vertices and m edges, $p \geqslant 1$ is an integer, and $\mathcal{N}$ is a monotone node search strategy of G_p, then there exists a monotone edge search strategy $\mathcal{S}$ of G_p such that*

$$\operatorname{cost}(\mathcal{S}) \leqslant \gamma(\mathcal{N}) + 2m(n+1).$$

Proof. By Lemma 1, there exists a node search strategy $\mathcal{N}$ of G_p such that the internal vertices of each path corresponding to an edge of G are guarded consecutively in $\mathcal{N}$. We construct an edge search strategy $\mathcal{S}$ of G_p. To this end we describe how a sequence of moves in $\mathcal{N}$, starting with $\mathcal{N}_i$, $i \in \{1, \ldots, |\mathcal{N}|\}$, and clearing the internal vertices of a path P that corresponds to an edge of G, is translated into the moves of $\mathcal{S}$.

Let uv be an edge in G with corresponding path P. Let x and y be adjacent to u and v, respectively, in P. If xu becomes clear in $\mathcal{N}_i$ and u is guarded at the end of $\mathcal{N}_i$, then first append to $\mathcal{S}$ a move that places a searcher on u, if only one searcher is at u in $\mathcal{S}$, and then append to $\mathcal{S}$ a move that slides a searcher along ux from u to x. If xu does not become cleared as a result of the move $\mathcal{N}_i$ then we place searchers on x in $\mathcal{S}$ to ensure that two searchers are present on x in $\mathcal{S}$, which introduces no cost in $\mathcal{S}$ as these are not clearing moves. By our choice of $\mathcal{N}$, the moves following $\mathcal{N}_i$ clear the remaining internal vertices of P. The corresponding moves of $\mathcal{S}$ slide a searcher from x to y. Note that as either the edge ux is cleared, or there are two searchers on x, these sliding moves clear the internal edges of P, and certainly no recontamination will occur, independently of whether ux is cleared. If, as a result of placing a searcher on y in $\mathcal{N}$, the edge yv becomes clear, then we slide the searcher occupying y from y to v in $\mathcal{S}$. Clearing the internal edges in $E(P) \setminus \{ux, vy\}$ gives the same cost both in $\mathcal{S}$ and in $\mathcal{N}$.

This leaves only the edges of G_p incident with a vertex of G to consider. In the worst case, there is an additional cost of $n+1$ in $\mathcal{S}$ for each edge of G_p adjacent to a vertex in $V(G)$. Since there are $2m$ such edges, we obtain $\operatorname{cost}(\mathcal{S}) \leqslant \gamma(\mathcal{N}) + 2m(n+1)$. □

Theorem 7. *For each graph G it holds $\operatorname{cost}_{\mathrm{m}}(G) = \operatorname{cost}(G)$.*

Proof. Let $p = 5n^3(n+1)$, and G be a graph with n vertices and m edges. Construct the graph G_p. Let $\mathcal{S}^{\mathrm{m}}(G)$ and $\mathcal{S}(G)$ be minimum cost monotone and non-monotone edge search strategies of G, respectively.

By Lemma 3, there exist the corresponding monotone and non-monotone edge search strategies $\mathcal{S}^{\mathrm{m}}(G_p)$ and $\mathcal{S}(G_p)$ of G_p such that

$$\operatorname{cost}\big(\mathcal{S}^{\mathrm{m}}(G_p)\big) = p \cdot \operatorname{cost}\big(\mathcal{S}^{\mathrm{m}}(G)\big) \quad \text{and} \quad \operatorname{cost}\big(\mathcal{S}(G_p)\big) = p \cdot \operatorname{cost}\big(\mathcal{S}(G)\big). \tag{5}$$

Moreover, $\mathcal{S}^{\mathrm{m}}(G_p)$ is a minimum cost monotone search strategy of G_p.

By Lemma 5, there exists a (possibly non-monotone) node search strategy $\mathcal{N}(G_p)$ of G_p with cost

$$\gamma\big(\mathcal{N}(G_p)\big) \leqslant \operatorname{cost}\big(\mathcal{S}(G_p)\big) + 2n^3(n+1). \tag{6}$$

Let $\mathcal{N}^{\mathrm{m}}(G_p)$ be a minimum cost monotone node search strategy of G_p. By the monotonicity property for the node search problem [4] we obtain that

$$\gamma\big(\mathcal{N}^{\mathrm{m}}(G_p)\big) \leqslant \gamma\big(\mathcal{N}(G_p)\big). \tag{7}$$

By Lemma 6, there exists a monotone edge search strategy $\widetilde{\mathcal{S}}^{\mathrm{m}}(G_p)$ of G_p with cost

$$\operatorname{cost}\big(\widetilde{\mathcal{S}}^{\mathrm{m}}(G_p)\big) \leqslant \gamma\big(\mathcal{N}^{\mathrm{m}}(G_p)\big) + 2m(n+1). \tag{8}$$

Since $m < n^3$, by (6), (7) and (8)

$$\operatorname{cost}\big(\widetilde{\mathcal{S}}^{\mathrm{m}}(G_p)\big) < \operatorname{cost}\big(\mathcal{S}(G_p)\big) + 5n^3(n+1). \tag{9}$$

Assume, by way of contradiction, that $\operatorname{cost}(\mathcal{S}(G)) \leqslant \operatorname{cost}(\mathcal{S}^{\mathrm{m}}(G)) - 1$. Then, by (5), $\operatorname{cost}(\mathcal{S}(G_p)) \leqslant \operatorname{cost}(\mathcal{S}^{\mathrm{m}}(G_p)) - p$, or $\operatorname{cost}(\mathcal{S}(G_p)) + p \leqslant \operatorname{cost}(\mathcal{S}^{\mathrm{m}}(G_p))$. Since $p = 5n^3(n+1)$, by (9), $\operatorname{cost}(\widetilde{\mathcal{S}}^{\mathrm{m}}(G_p)) < \operatorname{cost}(\mathcal{S}^{\mathrm{m}}(G_p))$, which contradicts the minimality of $\mathcal{S}^{\mathrm{m}}(G)$. Thus, $\operatorname{cost}(\mathcal{S}(G)) \geqslant \operatorname{cost}(\mathcal{S}^{\mathrm{m}}(G))$, and hence $\operatorname{cost}(\mathcal{S}(G)) = \operatorname{cost}(\mathcal{S}^{\mathrm{m}}(G)) = \operatorname{cost}_{\mathrm{m}}(G)$, as required. □

4. Approximation algorithm for general graphs

In this section we provide an approximation algorithm for calculating a minimum cost search strategy of an arbitrary graph $G = (V(G), E(G))$. For a node and edge search strategy $\mathcal{S}$ we define $B_b(\mathcal{S}_i)$ ($B_e(\mathcal{S}_i)$) to be the set of vertices that are guarded at the beginning (at the end, respectively) of a move $\mathcal{S}_i$, $i = 1, \ldots, |\mathcal{S}|$.

Lemma 8. *Let G be any graph. For each monotone node search strategy $\mathcal{N}$ of G_2 there exists an edge search strategy $\mathcal{S}$ of G such that* $\text{cost}(\mathcal{S}) \leqslant 4\gamma(\mathcal{N})$.

Proof. For each $uv \in E(G)$ denote by x_{uv} the node adjacent both to u and v in G_2. Given a monotone node search strategy $\mathcal{N}$ of G_2 we construct an edge search strategy $\mathcal{S}$ of G. To that end we perform the following for each $i = 1, \ldots, |\mathcal{N}|$.

Case 1. If $\mathcal{N}_i$ places a searcher on x_{uv}, $uv \in E(G)$, then we translate it into the moves in $\mathcal{S}$:
- (1a) if u or v has only one neighbor in G, then place a searcher σ on u or v, respectively, and slide σ along uv,
- (1b) if $|E_u| > 1$, $|E_v| > 1$ and the only contaminated edge in E_u is ux_{uv} or the only contaminated edge in E_v is vx_{uv}, then slide in $\mathcal{S}$ the searcher present at u or v, respectively, along uv,
- (1c) if at least two edges in E_u and at least two edges in E_v are contaminated in $\mathcal{N}$, then place two searchers on u, respectively, and slide one of them from u to v.

Case 2. if $\mathcal{N}_i$ places a searcher on $v \in V(G)$, then in $\mathcal{S}$ we do nothing.

Then, before proceeding to $i+1$, repeat the following as long as possible: if a searcher σ is present on a clear vertex or if another searcher is at v, then add to $\mathcal{S}$ a move that removes σ from v.

One can prove by an induction on $i \in \{1, \ldots, |\mathcal{N}|\}$ that a searcher is placed on x_{uv} in one of the moves $\mathcal{N}_1, \ldots, \mathcal{N}_i$ if and only if uv has been cleared by the corresponding moves of $\mathcal{S}$. This implies that $\mathcal{S}$ is a valid edge search strategy of G.

We prove by induction on the number of moves in $\mathcal{N}$, that if $\mathcal{S}_{j_i}$ is the last move among the moves of $\mathcal{S}$ corresponding to $\mathcal{N}_i$, $i \in \{1, \ldots, |\mathcal{N}|\}$, then for each $u' \in V(G)$ it holds

$$u' \in B_e(\mathcal{S}_{j_i}) \quad \Rightarrow \quad \big(u' \in B_e(\mathcal{N}_i) \text{ or } x_{u'v'} \in B_e(\mathcal{N}_i)\big) \tag{10}$$

for some neighbor v' of u' in G. Suppose that (10) holds for each $i \in \{1, \ldots, |\mathcal{N}| - 1\}$ and we consider the move $\mathcal{N}_{i+1}$. It is enough to analyze the case when $\mathcal{N}_{i+1}$ places a searcher on x_{uv} for some $uv \in E(G)$. Observe that

$$B_e(\mathcal{N}_{i+1}) \setminus \{u, v, x_{uv}\} = B_e(\mathcal{N}_i) \setminus \{u, v, x_{uv}\} \tag{11}$$

and

$$B_e(\mathcal{S}_{j_{i+1}}) \setminus \{u, v\} = B_e(\mathcal{S}_{j_i}) \setminus \{u, v\}. \tag{12}$$

Hence, if step (1c) has been performed for $\mathcal{N}_{i+1}$, then (10) holds for $i+1$. Suppose for a contradiction that (10) does not hold when step (1a) or (1b) has been performed. By (11) and (12), it fails for $u' = u$ or $u' = v$. We consider the former case, as the other one is symmetric. Hence, $u \in B_e(\mathcal{S}_{j_{i+1}})$ and $u \notin B_e(\mathcal{N}_i)$ and $x_{uv'} \notin B_e(\mathcal{N}_{i+1})$ for each neighbor v' of u in G. The fact that $u \in B_e(\mathcal{S}_{j_{i+1}})$ implies that an edge $uv' \in E_u$ is not cleared at the end of $\mathcal{S}_{j_{i+1}}$ for some neighbor v' of u in G. Hence, $x_{uv'}$ is contaminated at the end of $\mathcal{N}_{i+1}$. This gives the desired contradiction and proves the claim.

Eq. (10) implies

$$\big|B_e(\mathcal{S}_{j_i})\big| \leqslant 2\big|B_e(\mathcal{N}_i)\big| \quad \text{for each } i = 1, \ldots, |\mathcal{N}|. \tag{13}$$

By construction, for each $i = 1, \ldots, |\mathcal{N}|$ there exists exactly one integer $t_i \in \{j_{i-1} + 1, \ldots, j_i\}$ (take $j_0 = 0$) such that $\mathcal{S}_{t_i}$ is a clearing move. Note that

$$|\mathcal{S}_{t_i}| \leqslant 2\big|B_e(\mathcal{N}_i)\big| + 2 \quad \text{for each } i = 1, \ldots, |\mathcal{N}|. \tag{14}$$

Indeed, at most two searchers are removed from u and v after clearing uv, which implies $|\mathcal{S}_{t_i}| \leqslant |B_e(\mathcal{S}_{j_i})| + 2$ and therefore (14) follows from (13). Eq. (14) completes the proof. □

For the analysis of our approximation algorithm we will need a lower bound for the edge search cost a graph.

Lemma 9. *For each graph G we have* $\text{cost}(G) \geqslant \gamma(G)/3$.

Proof. Given an edge search strategy $\mathcal{S}$ of G we construct a node search strategy $\mathcal{N}$ of G by translating each move $\mathcal{S}_i$ that clears an edge uv of G into two moves in $\mathcal{N}$ of placing the searchers on u and v, respectively (if u and/or v is already guarded, then we do not place a second searcher on the vertex). If at the end of $\mathcal{S}_i$, $i \in \{1, \ldots, |\mathcal{S}|\}$, the vertex v (respectively u) is not guarded, then we remove the corresponding searcher from v (respectively u) in the corresponding

move of $\mathcal{N}$. In this way after the corresponding moves of $\mathcal{S}$ and $\mathcal{N}$, the sets of guarded vertices are equal. Note that the searchers are removed from the vertices of G by $\mathcal{N}$ not necessarily immediately when they are not needed to guard them, which implies that in this node search strategy $\mathcal{N}$ we do include those searchers while calculating its cost. Hence, the cost of $\mathcal{N}$ is an upper bound on $\gamma(G)$.

Assume that $\mathcal{S}_i$ slides a searcher from u to v to clear uv and uses s searchers in this move. Let $\mathcal{N}_j$, $\mathcal{N}_{j+1}$ be the two corresponding clearing moves in $\mathcal{N}$ (placing searchers at u and v, respectively), which result in clearing uv as well. (If less than two moves of $\mathcal{N}$ correspond to $\mathcal{S}_i$, then the analysis is simpler and we omit it.) We have that $\mathcal{N}_j$ uses s searchers and $\mathcal{N}_{j+1}$ uses $s+1$ searchers. This implies that $(|\mathcal{N}_j| + |\mathcal{N}_{j+1}|)/|\mathcal{S}_i| \leqslant 3$. □

Recall that the *profile* of a graph G, $p(G)$, is the minimum number of edges in an interval supergraph of G, i.e.

$$p(G) = \min\{|E(H)|\colon H \text{ is an interval supergraph of } G\}.$$

Theorem 10. *(See [4].) For each simple graph G it holds $p(G) = \gamma(G)$.* □

The problem of deciding whether $p(G) \leqslant k$ for given G and k is NP-complete [4,6]. We also have the following.

Theorem 11. *(See [10].) There exists an $O(\log n)$-approximation polynomial-time algorithm for profile minimization, where $n = |V(G)|$.*

Our approximation algorithm for finding minimum cost edge search strategy can be described as follows:

Step 1: Given G, compute the graph G_2.
Step 2: Find a node search strategy $\mathcal{N}$ of G_2. To this end we use the $O(\log|V(G_2)|)$-approximation algorithm for minimizing the profile of G_2. By Theorems 10 and 11, this gives a $O(\log|V(G_2)|)$-approximate solution $\mathcal{N}$ to the minimum cost node searching problem for G_2.
Step 3: Translate $\mathcal{N}$ into edge search strategy $\mathcal{S}$ of G as described in the proof of Lemma 8. (Note that $\text{cost}(\mathcal{S}) \leqslant 4\gamma(\mathcal{N})$.)

By the discussion above, $\gamma(\mathcal{N}) \leqslant cp(G_2)\log|V(G_2)| = c\gamma(G_2)\log|V(G_2)|$, where c is a constant. Since $|V(G_2)| = |V(G)| + |E(G)| \leqslant |V(G)|^2$, we obtain by Lemma 8 that $\text{cost}(\mathcal{S}) \leqslant 4\gamma(\mathcal{N}) \leqslant 8c\gamma(G_2)\log|V(G)|$. By Lemma 9 (applied for G_2),

$$\frac{\text{cost}(\mathcal{S})}{\text{cost}(G)} \leqslant 8c\frac{\gamma(G_2)}{\text{cost}(G)}\log|V(G)| \leqslant 24c\frac{\text{cost}(G_2)}{\text{cost}(G)}\log|V(G)|.$$

By Lemma 3, $\text{cost}(G_2) = 2\text{cost}(G)$. This gives the following.

Theorem 12. *There exists a polynomial-time $O(\log n)$-approximation algorithm for finding minimum cost edge search strategy for any graph G, where n is the number of vertices in G.*

5. Minimum cost search for complete graphs

In this section we give a formula for the edge search cost of an n-vertex complete graph, denoted by K_n. Our proof is constructive, i.e., we provide an algorithm for computing a minimum cost search strategy. Let $V(K_n) = \{v_1, \ldots, v_n\}$. In the following we assume that vertices are labelled in the order they are first visited; that is, without loss of generality, that in each search strategy of the complete graph when a searcher reaches a vertex v_i for the first time (i.e. v_i becomes guarded), then all searchers currently in the graph are on vertices with index at most i, and all other vertices $(v_{i+1}, \ldots, v_n)$ are contaminated.

Naively, two strategies present themselves as methods for clearing K_n in an attempt to minimize the cost. The first would be to clear K_n "clique by clique", where the graph induced by cleared edges induces a complete graph on v_1 and v_2, then v_1, v_2, and v_3, then v_1, v_2, v_3, v_4, and so on, until the graph is cleared. While this allows many edges to be cleared cheaply early in the strategy, it later becomes expensive. Alternatively, K_n may be cleared "vertex by vertex", where searchers are placed on all vertices, and then v_1 is cleared, then v_2, then v_3, and so on. In such a strategy, clearing early vertices is expensive, but later vertices are cheap. In general, it turns out that neither of these search strategies minimizes the cost. In fact, a minimum cost search strategy that we present in this section combines these two ideas, first clearing clique by clique, and eventually ending vertex by vertex.

Lemma 13. *There exists a minimum cost monotone search strategy $\mathcal{S}$ of K_n, $n \geqslant 4$, such that the vertices of K_n become clear in the same order as they become guarded.*

Proof. Since we know by Theorem 7 that recontamination does not reduce cost, we assume without loss of generality that all search strategies we consider are monotone. For any search strategy $\mathcal{S}$ let $r(\mathcal{S})$ be the minimum index i such that v_i is not clear at the end of a move $\mathcal{S}_p$ for some $p \in \{1, \ldots, |\mathcal{S}|\}$ and v_j, for some $j > i$, becomes clear in the move $\mathcal{S}_p$, and let $r(\mathcal{S}) = n$ if no such index i exists. Note that we need to prove that there exists a minimum cost search strategy $\mathcal{S}$ with $r(\mathcal{S}) = n$. Suppose for a contradiction that such a search strategy does not exist and let $\mathcal{S}$ be a minimum cost search strategy with the maximum value of $r(\mathcal{S})$. Denote for brevity $r(\mathcal{S}) = i$, and let $j > i$ be such that v_j becomes cleared in $\mathcal{S}$ prior to v_i. We construct a search strategy $\mathcal{S}'$ with the same cost as $\mathcal{S}$ and $r(\mathcal{S}') > r(\mathcal{S})$, which will give a desired contradiction.

Let p be the smallest integer such that at the end of move $\mathcal{S}_p$ all edges in E_{v_j} are clear. Then all moves in $\mathcal{S}'$ are the same as in $\mathcal{S}$ with the following exceptions. If $\mathcal{S}_l$, where $l \leqslant p$, clears an edge $v_j v_k$ (for $1 \leqslant k \leqslant n$, $k \neq i$), then let $\mathcal{S}'_l$ clear $v_i v_k$. Similarly, if $\mathcal{S}_l$, where $l > p$, clears an edge $v_i v_k$, then let $\mathcal{S}'_l$ clear $v_j v_k$. Further, to ensure that these moves are possible, any moves in $\mathcal{S}$ before p that place a searcher on v_j that is subsequently used to clear an edge $v_j v_k$ will be replaced in $\mathcal{S}'$ by moves that place a searcher on v_i, and any move in $\mathcal{S}$ after p that places a searcher on v_i that is subsequently used to clear an edge $v_i v_k$ is replaced in $\mathcal{S}'$ by a move that places a searcher on v_j. A similar adjustment is made for removing searchers from v_i and v_j, again depending on their occurrence before or after p.

Now we prove that $\mathcal{S}'$ is a valid search strategy of K_n. Since we only change the order of moves in $\mathcal{S}$ to obtain $\mathcal{S}'$, $|\mathcal{S}'| = |\mathcal{S}|$ and it is enough to prove that no recontamination occurs in $\mathcal{S}'$. Suppose, by way of contradiction, that an edge in E_{v_k}, $k \in \{1, \ldots, n\}$, becomes recontaminated. Since the vertices in $V(K_n) \setminus \{v_i, v_j\}$ become guarded and then the searchers guarding them are removed in the same order in $\mathcal{S}$ and in $\mathcal{S}'$, we know that a recontamination in $\mathcal{S}'$ occurs by removing a searcher from v_k while either $v_k v_j$ or $v_k v_i$ is contaminated. However, if $v_k v_j$ ($v_k v_i$) is the edge that causes recontamination in $\mathcal{S}'$, then by the definition of $\mathcal{S}'$, $v_k v_i$ (respectively, $v_k v_j$) will cause recontamination in $\mathcal{S}$, a contradiction, since $\mathcal{S}$ is monotone.

Thus, since we have changed the order in which vertices become cleared, but introduced no new searchers in each move of $\mathcal{S}'$, we know that $|\mathcal{S}'| = |\mathcal{S}|$, and hence that $\text{cost}(\mathcal{S}) = \text{cost}(\mathcal{S}')$. Further, $r(\mathcal{S}) = i$ and $r(\mathcal{S}') \geqslant i + 1$. This last statement follows from the fact that v_i is clear at the end of the move $\mathcal{S}'_p$, while some edges in E_{v_j} are contaminated at the end of $\mathcal{S}'_p$ because there are contaminated edges in E_{v_i} in $\mathcal{S}$ at the end of $\mathcal{S}_p$. □

In order to simplify the analysis we define

$$g(\mathcal{S}_i) = \begin{cases} 1 + |B_b(\mathcal{S}_i) \cup B_e(\mathcal{S}_i)|, & \text{if } \mathcal{S}_i \text{ is a clearing move,} \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

$i = 1, \ldots, |\mathcal{S}|$. We obtain the following.

Lemma 14. *If $\mathcal{S}$ is a monotone search strategy for K_n, then*

$$\text{cost}(\mathcal{S}) \geqslant \sum_{1 \leqslant i \leqslant |\mathcal{S}|} g(\mathcal{S}_i) - 2(n-1).$$

Proof. We distinguish three (not necessarily distinct) types of clearing moves $\mathcal{S}_i$ that may occur in a search strategy $\mathcal{S}$:

Type 1: a searcher σ slides along an edge $v_i v_j$ while two additional searchers guard v_i and v_j,
Type 2: a searcher σ slides from v_i to v_j while v_i is not guarded,
Type 3: a searcher σ slides from v_i to v_j while v_j is not guarded.

The number of searchers used in a move $\mathcal{S}_k$ of Type 1 is at least $g(\mathcal{S}_k)$, because the vertices in $B_b = B_e$ are guarded while an additional searcher clears an edge. A move of Type 2 occurs when all edges in E_{v_i} except $v_i v_j$ are cleared at the beginning of $\mathcal{S}_k$. Thus, at the end of this move v_i is clear. This happens at most once for each $i = 1, \ldots, n-1$, and does not happen for v_n, the vertex that becomes clear last. Moreover, for a move of Type 2, the number of searchers used is at least $g(\mathcal{S}_k) - 1$ if the particular move is not of Type 3 simultaneously. Analogously, a move $\mathcal{S}_k$ of Type 3 is an event when a searcher reaches v_j for the first time. Each vertex, except for v_1, may get guarded as a result of such move, and the number of searchers used in $\mathcal{S}_k$ is $g(\mathcal{S}_k) - 1$ if, again, the move is not of Type 2 simultaneously. Finally, a move can be of Type 2 and 3 and, by the definition, such a case may happen once during a search strategy, and the number of searchers in use is then $g(\mathcal{S}_k) - 2$. This proves that the number of clearing moves $\mathcal{S}_k$ that use $g(\mathcal{S}_k) - 1$ searchers is at most $2(n-2)$, and there may exist at most one clearing move using $g(\mathcal{S}_k) - 2$ searchers. □

The consequence of Lemma 14 is that if we are able to find a search strategy $\mathcal{S}$ such that $s = \sum_{1 \leqslant i \leqslant |\mathcal{S}|} g(\mathcal{S}_i)$ is minimum and $\text{cost}(\mathcal{S}) = s - 2(n-1)$, then the cost of $\mathcal{S}$ is minimum, i.e., $\text{cost}(\mathcal{S}) = \text{cost}(G)$.

Consider a minimum cost monotone search strategy $\mathcal{S}$ of K_n that satisfies the condition of Lemma 13. Assume that v_i is guarded for the first time at the end of move $\mathcal{S}_{l_i}$, $i = 1, \ldots, n$. By assumption, $l_i < l_{i+1}$ for each $i = 1, \ldots, n-1$. Similarly,

let $\mathcal{S}_{t_i}$ be the first move at the end of which v_i is clear, $i = 1, \ldots, n$. By the choice of $\mathcal{S}$, $t_i < t_{i+1}$ for each $i = 1, \ldots, n-1$. Since the earliest the first vertex can be cleared is when the same moment a searcher may first enter v_n, $l_n \leqslant t_1$. For brevity denote $l_{n+1} = t_1$.

Given a monotone search strategy $\mathcal{S}$ of K_n, define $X_i(\mathcal{S})$ to be the set of edges $v_j v_i$, $j < i$, cleared while none of the vertices $v_{i+1}, \ldots, v_n$ is guarded, $i = 1, \ldots, n$. Let $Y_i(\mathcal{S})$, $i = 1, \ldots, n$, be the set of edges in E_{v_i} still contaminated when the vertex v_{i-1} becomes clear, and the vertices $v_1, v_2, \ldots, v_{i-2}$ remain clear and unoccupied. In the latter case, by Lemma 13, the vertices $v_{i+1}, \ldots, v_n$ are guarded while clearing the edges in $Y_i(\mathcal{S})$.

Lemma 15. *There exists a minimum cost monotone search strategy $\mathcal{S}$ of K_n, $n \geqslant 2$, such that for all $2 \leqslant i \leqslant n$ and $1 \leqslant k \leqslant i-1$,*

$$v_k v_i \in X_i(\mathcal{S}) \cup Y_k(\mathcal{S}).$$

Proof. Let $\mathcal{S}$ be a minimum cost monotone search strategy of K_n in which the vertices become clear in the same order as they become guarded. By Lemma 13, such a search strategy exists. We prove that an edge $v_k v_i$, $k < i$, is cleared in a move $\mathcal{S}_j$, where

$$\mathcal{S}_j \in \{\mathcal{S}_{l_i}, \ldots, \mathcal{S}_{l_{i+1}-1}\} \cup \{\mathcal{S}_{t_{k-1}+1}, \ldots, \mathcal{S}_{t_k}\} \quad \text{for each } i = 2, \ldots, n,$$

where $t_0 = t_1 - 1$.

By the definition, $j \geqslant l_i$. Since $k < i$, $j \neq l_{i+1}$. If $j \in \{l_{i+1}, \ldots, t_1 - 1\}$, then $g(\mathcal{S}_j) > i+1$. Instead, we alter $\mathcal{S}$, so that clearing $v_k v_i$ follows the move $\mathcal{S}_{l_i}$. This results in $g(\mathcal{S}_{l_i+1}) = i+1$, and gives a strategy of lesser cost, which contradicts the minimality of $\mathcal{S}$. If $k = 1$, then $j \geqslant t_1 = t_{k-1} + 1$. If $k > 1$, then, by definition, $j \neq t_1$. Hence, if $k > 1$ and $j \in \{t_1 + 1, \ldots, t_{k-1} - 1\}$, then $g(\mathcal{S}_j) > n - k + 2$. But if we alter $\mathcal{S}$ so that $v_k v_i$ is cleared as move $\mathcal{S}_{t_{k-1}+1}$, this gives $g(\mathcal{S}_{t_{k-1}+1}) = n - k + 2$, again contradicting the minimality of $\mathcal{S}$. Since we know that $v_k v_i$ must be cleared when v_k becomes clear, $j \leqslant t_k$, and the result follows. □

Lemma 16. *There exists a minimum cost monotone search strategy $\mathcal{S}$ of K_n, $n \geqslant 2$, such that $X_1(\mathcal{S}) = Y_1(\mathcal{S}) = Y_n(\mathcal{S}) = \emptyset$, and*

$$X_i(\mathcal{S}) = \{v_1 v_i, \ldots, v_{i-1} v_i\}, \quad i = 2, \ldots, \lfloor (n+1)/2 \rfloor, \tag{16}$$

$$X_i(\mathcal{S}) = \{v_1 v_i, \ldots, v_{n-i+1} v_i\}, \quad i = \lfloor (n+1)/2 \rfloor + 1, \ldots, n, \tag{17}$$

$$Y_i(\mathcal{S}) = \{v_{n-i+2} v_i, \ldots, v_n v_i\}, \quad i = 2, \ldots, \lfloor (n+1)/2 \rfloor, \tag{18}$$

$$Y_i(\mathcal{S}) = \{v_{i+1} v_i, \ldots, v_n v_i\}, \quad i = \lfloor (n+1)/2 \rfloor + 1, \ldots, n-1. \tag{19}$$

Proof. Let $\mathcal{S}$ be a monotone minimum cost search strategy of K_n. By Lemmas 13 and 15, for $k < i$, $v_k v_i \in X_i(\mathcal{S})$ or $v_k v_i \in Y_k(\mathcal{S})$. That is, either $v_k v_i$ is cleared in one of the moves $\mathcal{S}_{l_i}, \ldots, \mathcal{S}_{l_{i+1}-1}$, or $v_k v_i$ is cleared in one of the moves $\mathcal{S}_{t_{k-1}+1}, \ldots, \mathcal{S}_{t_k}$. Let $\mathcal{S}_j$ be the move in which $v_k v_i$ is cleared.

Let first $i < n$. If $v_k v_i \in X_i(\mathcal{S})$, then $\mathcal{S}_j$ uses $i+1$ searchers, and if $v_k v_i \in Y_k(\mathcal{S})$, then $\mathcal{S}_j$ uses $n - k + 2$ searchers, as in the proof of Lemma 15. Since $\mathcal{S}$ is a minimum cost search strategy of K_n, $v_k v_i \in X_i(\mathcal{S})$ if and only if $i + 1 \leqslant n - k + 2$. This implies that

$$X_i(\mathcal{S}) = \{v_k v_i \colon k < i,\ i + 1 \leqslant n - k + 2\}. \tag{20}$$

Hence, (16) and (17) follow from (20). Since, by Lemma 15, $\bigcup_{1 \leqslant i \leqslant n} Y_i(\mathcal{S}) = E(K_n) \setminus \bigcup_{1 \leqslant i \leqslant n} X_i(\mathcal{S})$, we obtain

$$Y_i(\mathcal{S}) = \{v_k v_i \colon k > i,\ i > n - k + 1\}, \tag{21}$$

which leads to (18) and (19).

Finally note that $Y_n(\mathcal{S}) \neq \emptyset$ by definition, and $X_n(\mathcal{S}) = E_{v_n} \setminus \bigcup_{k<n} Y_k(\mathcal{S}) = \{v_1 v_n\}$. □

Now we define a search strategy $\mathcal{S}$ that satisfies the conditions from Lemma 16. Hence, we conclude that $\mathcal{S}$ is of minimum cost. In the pseudo-code below the instruction clear(a, b), $1 \leqslant a < b \leqslant n$, stands for the sequence of three actions:

1. If more than one edge in E_{v_a} is contaminated, then place a searcher σ on v_a. Otherwise let σ be the only searcher present at v_a.
2. Slide σ from v_a to v_b.
3. If v_b is clear, then remove all searchers from v_b. If v_b is not clear and two searchers occupy v_b, then remove σ from v_b.

The minimum cost search strategy $\mathcal{S}$ for K_n, $n \geqslant 4$, is constructed as follows:

```
Place a searcher on v_1.
for i = 2 to ⌊(n+1)/2⌋ do
    for j = 1 to i − 1 do
        clear(j, i)
for i = ⌊(n+1)/2⌋ + 1 to n do
    for j = 1 to n − i + 1 do
        clear(j, i)
for i = 2 to ⌊(n+1)/2⌋ do
    for j = n − i + 2 to n do
        clear(i, j)
for i = ⌊(n+1)/2⌋ + 1 to n − 1 do
    for j = i + 1 to n do
        clear(i, j)
```

We finish this section by calculating $\mathrm{cost}(\mathcal{S})$ which, due to the optimality of $\mathcal{S}$, leads to a formula for $\mathrm{cost}(K_n)$. By Lemma 15, we know that every edge is in either $X_i(\mathcal{S})$ or $Y_i(\mathcal{S})$ for some i. Again, we consider the move $\mathcal{S}_j$ where an edge v_iv_k $(k < i)$ is cleared.

If $\mathcal{S}_j$ clears $v_iv_k \in X_i(\mathcal{S})$, $k < i$, then, as in the proof of Lemma 15, $g(\mathcal{S}_j) = i+1$ and if $\mathcal{S}_j$ clears $v_iv_k \in Y_i(\mathcal{S})$, $k > i$, then $g(\mathcal{S}_j) = n-i+2$. Thus, by Lemma 16,

$$\sum_{j=1,\ldots,|\mathcal{S}|} g(\mathcal{S}_j) = \sum_{i=1}^{\lfloor (n+1)/2 \rfloor} (i-1)(i+1) + \sum_{i=\lfloor \frac{n+1}{2} \rfloor + 1}^{n} (n-i+1)(i+1) + \sum_{i=2}^{\lfloor (n+1)/2 \rfloor} (i-1)(n-i+2) + \sum_{i=\lfloor \frac{n+1}{2} \rfloor + 1}^{n-1} (n-i)(n-i+2).$$

Note that in this strategy, the move that clears the edge $v_1v_n \in X(\mathcal{S})$ leaves all edges in E_{v_1} cleared and a searcher reaches v_n for the first time during the search strategy. That is, $\mathcal{S}_{l_n} = \mathcal{S}_{t_1}$. Moreover, $g(\mathcal{S}_{l_n}) = n+1$ and the number of searchers used in $\mathcal{S}_{l_n}$ is $n-1$. Since the moves $\mathcal{S}_{l_j}$, $j = 2, \ldots, n-1$ and $\mathcal{S}_{t_j}$, $j = 2, \ldots, n-1$ use $g(\mathcal{S}_j) - 1$ searchers, we obtain that there are exactly $2(n-2)$ clearing moves $\mathcal{S}_j$ using $g(\mathcal{S}_j) - 1$ searchers and one move (clearing the edge v_1v_n) using $g(\mathcal{S}_j) - 2$ searchers. The remaining clearing moves for other edges use $g(\mathcal{S}_j)$ searchers. Therefore, $\mathrm{cost}(\mathcal{S}) = \sum_{j=1,\ldots,|\mathcal{S}|} g(\mathcal{S}_j) - 2(n-1)$, and, by Lemma 14, we obtain the following for the graph K_n.

Theorem 17. *If $n \geqslant 4$, then*

$$\mathrm{cost}(K_n) = \begin{cases} \frac{1}{4}n^3 + \frac{5}{8}n^2 - \frac{11}{4}n + \frac{15}{8}, & \text{if } n \text{ is odd}, \\ \frac{1}{4}n^3 + \frac{5}{8}n^2 - \frac{11}{4}n + 2, & \text{if } n \text{ is even}. \end{cases}$$

Moreover, there exists a minimum cost monotone search strategy of K_n that uses $\mathrm{s}(K_n) = n$ searchers. That is, an ideal search strategy for K_n exists.

6. Ideal search strategies do not exist for all graphs

Having proved that ideal search strategies do exist for complete graphs, we prove in this section that not all graphs have this property.

In the following, the vertices of each clique K_i, $i > 0$, are numbered $0, \ldots, i-1$. We say that two cliques K_j, K_l, $j \leqslant l$, are *connected* if there exist l edges between their vertices placed as follows. The i-th of those edges connects the vertex number $((i-1) \bmod j)$ in K_j with the $(i-1)$-st vertex of K_l. We fix integers $k \geqslant 1$ and $p \geqslant 1$, though the construction is valid for each $k \geqslant 1$ and for each $p \geqslant 1$. However, to obtain Theorem 20 we will later be interested in considering "sufficiently large" values for p.

Construct a graph B by taking p copies of K_{3k}, which we will refer to as the *medium cliques* of B, and connect them in series so that the i-th clique is connected to the $(i+1)$-st clique, $i = 1, \ldots, p-1$. In addition, connect the p-th clique of the chain to a clique K_k, which we call the *small clique* of B, and finally connect the small clique with a copy of K_{4k}, the *large clique* of B. We call such a subgraph B a *branch*. Define $G(k, p)$ to be a graph consisting of four branches B_1, B_2, B_3, and B_4 and an additional clique K_{2k}, called the *central clique* of $G(k, p)$, where the first medium clique in each branch is connected to the central clique. Fig. 1 depicts $G(2, 3)$.

Lemma 18. *If $k \geqslant 1$ and $p \geqslant 1$ are any integers, then $\mathrm{s}(G(k, p)) \leqslant 5k + 1$.*

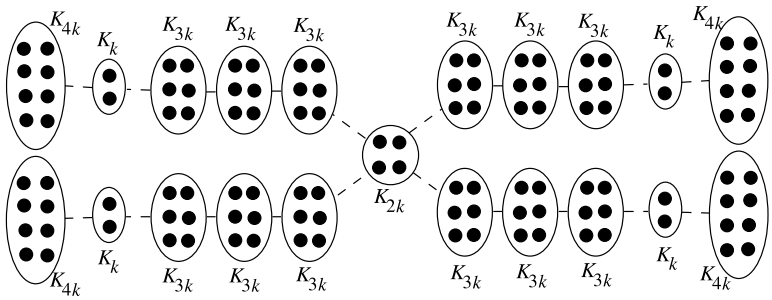

Fig. 1. The graph $G(k, p)$ with $k = 2$ and $p = 3$.

Proof. We sketch a search strategy that uses $5k + 1$ searchers. In this strategy, we will clear each branch "clique by clique", in that, for a given clique, after obtaining a single searcher on each vertex of the clique, then use an additional searcher to clear all edges of that clique, before proceeding to the next clique.

Specifically, when a clique K_j is clear and the strategy proceeds to clear the only uncleared clique $K_{j'}$ connected to K_j, then either $j' \leqslant j$ or $j' > j$. In the former case, the searcher on vertex i in K_j slides along the (contaminated) edge to vertex i in $K_{j'}$, for $0 \leqslant i \leqslant j' - 1$. Then, those searchers remaining on K_j are removed. This ends with all of the vertices of $K_{j'}$ occupied, and all edges between K_j and $K_{j'}$ cleared.

If $j' > j$ we first place $j' - j$ searchers on the vertices of K_j so that there is one searcher on each vertex of K_j for each contaminated edge between that vertex and the vertices of $K_{j'}$. Then, slide each of the j' searchers on K_j along a contaminated edge incident with its vertex to $K_{j'}$, clearing that edge in the process. Again, this ends with all of the vertices of $K_{j'}$ occupied, and all of the edges between K_j and $K_{j'}$ cleared.

First we clear the large clique of B_1 and then the small clique of B_1. While the k vertices of the small clique of B_1 are guarded, clear the branch B_2 by clearing its cliques consecutively, starting with the large clique, then the small clique, then the medium cliques, finally ending with the medium clique connected to the central clique of $G(k, p)$. Then, clear the central clique. At this point, the branch B_2 is clear.

Leaving $2k$ searchers guarding the vertices of the central clique, clear the medium cliques of B_1, starting at the small clique and progressing sequentially along the medium cliques until the central clique is reached. This results in both branches B_1 and B_2 being cleared, and the only clique that is guarded now is the central clique. To clear B_3 and B_4 we reverse the procedure of clearing B_1 and B_2.

It is straightforward to verify that this search strategy is valid and uses $5k + 1$ searchers. □

Lemma 19. *For all $q \geqslant 0$ there exist integers k, p, and a search strategy $\mathcal{S}$ of $G(k, p)$ using $6k + 1$ searchers such that for each search strategy $\mathcal{S}'$ of $G(k, p)$ using $s(G(k, p))$ searchers it holds* $\mathrm{cost}(\mathcal{S}') - \mathrm{cost}(\mathcal{S}) \geqslant q$.

Proof. Set k to be any integer greater than 1, and let p be any integer that satisfies

$$p \geqslant 2\big(q + (6k + 1)^3\big)/\big(k(k - 1)\big). \tag{22}$$

Let $\mathcal{S}'$ be of minimum cost among all search strategies using $s(G(k, p))$ searchers. By Lemma 18, $s(G(k, p)) \leqslant 5k + 1$. Also, $s(K_n) = n$ for each $n \geqslant 4$ [12]. Hence, while clearing the edges of a large clique, at most $k + 1$ searchers can be used for guarding the vertices of other cliques. Thus, at most two large cliques can be cleared before clearing a medium clique or the central clique.

By symmetry, $\mathcal{S}'$ must clear the two remaining large cliques after clearing all medium cliques and the central clique, which means that $\mathcal{S}'$ clears two chains of medium cliques, say in branches B_3 and B_4 before clearing the large cliques in those branches. Since two medium cliques cannot be guarded simultaneously, without loss of generality let $\mathcal{S}'_a$ be the move after which the small clique of B_3 is guarded and all the medium cliques in B_3 are clear, but none of the cliques of branch B_4 are clear. Then, $\mathcal{S}'$ clears the remaining part of the graph. In particular, note that all the medium cliques of B_4 are cleared while the small clique of B_3 remains guarded.

Now we define a search strategy $\mathcal{S}$ of $G(k, p)$ that uses $(6k+1)$ searchers, and then we argue that $\mathrm{cost}(\mathcal{S}') - \mathrm{cost}(\mathcal{S}) \geqslant q$. Let $\mathcal{S}_i = \mathcal{S}'_i$ for each $i = 1, \ldots, a$. Following the move $\mathcal{S}_a$, $\mathcal{S}$ clears the contaminated edges of the small clique of B_3, if any, and then the edges of the large clique of B_3. This uses $4k + 1$ searchers, as well as the additional $2k$ searchers that remain guarding the central clique. Then, $\mathcal{S}$ clears the remaining edges of $G(k, p)$ in the same order as they are cleared by $\mathcal{S}'$.

The additional cost introduced in $\mathcal{S}$ is the cost of clearing the edges of the small and large cliques of B_3, together with edges between them, while guarding the central clique, where each move requires at most $6k + 1$ searchers. The number of such moves is the number of edges cleared, i.e., the edges in K_k, K_{4k} and the edges connecting the large and small cliques,

that is, $4k(4k-1)/2+k(k-1)/2+4k$. Each of the remaining moves of $\mathcal{S}$ requires no more searchers than the corresponding move of $\mathcal{S}'$. Thus, the additional cost introduced in $\mathcal{S}$ is at most

$$(6k+1)\big(4k(4k-1)/2+k(k-1)/2+4k\big) \leqslant (6k+1)^3. \tag{23}$$

On the other hand, the additional cost of $\mathcal{S}'$ with respect to $\mathcal{S}$ is the cost of guarding the small clique of branch B_3 while clearing the medium cliques of branch B_4, which is at least

$$pk(k-1)/2. \tag{24}$$

Then the difference in the costs of these strategies is at least the difference in expressions (23) and (24),

$$\operatorname{cost}\big(\mathcal{S}'\big) - \operatorname{cost}(\mathcal{S}) \geqslant pk(k-1)/2 - (6k+1)^3.$$

Thus, by (22), $\operatorname{cost}(\mathcal{S}') - \operatorname{cost}(\mathcal{S}) \geqslant q$, as required. □

By choosing sufficiently large p relative to q and k, Lemma 19 gives us Theorem 20.

Theorem 20. *For each constant $q \geqslant 0$ there exists a graph G such that $\operatorname{cost}(\mathcal{S}) - \operatorname{cost}(G) \geqslant q$ for each search strategy of G that uses $\mathrm{s}(G)$ searchers.*

7. Conclusions

The most obvious open question with regard to the edge search cost of a graph involves its complexity. For a given graph G, is determining $\operatorname{cost}(G) \leqslant k$ NP-complete? We have laid the groundwork for this problem by showing that cost is monotone. Intuitively, since determining if $\gamma(G) \leqslant k$ is NP-complete (as is computing the edge and node search numbers) this parameter should also be NP-complete. Of particular interest would be to determine families of graphs for which the edge search cost can be computed efficiently; trees are of particular interest.

We have shown that an $O(\log n)$-approximation algorithm exists, but it similarly remains open as to whether this is best possible. A strongly related question is whether there is a known graph parameter that is the same as (or close to) cost, as profile is to node search cost.

Movement in edge searching consists of two dissimilar modes. A searcher either slides or jumps. The cost function we have examined deals with a subset of the sliding moves; particularly, those moves that clear an edge. Obviously, these moves are fundamental to any search strategy, while jumping is not. In any "real world scenario" however, jumping would be very expensive. This motivates two related problems: first, what can be said about the minimum cost of a monotone internal search (in which jumping is not allowed), and second, what is the minimum costs of a search in which there was a (proportionally higher?) cost for jumping?

The result of Theorem 20 tells us that in general, we may minimize the edge search number or the edge search cost, but not both. However, for complete graphs in particular Theorem 17 tells us that ideal search strategies exist. More generally, we wonder for which graphs ideal strategies exist. Moreover, we know that for complete graphs, the edge search cost is $O(n^3)$, but the node search cost is $O(n^2)$. However, in our discussion of monotonicity, we introduce graphs G_p for which the both of these costs are quite close for large enough p. For what graphs are these two costs essentially the same?

References

[1] D. Bienstock, P. Seymour, Monotonicity in graph searching, J. Algorithms 12 (2) (1991) 239–245.
[2] R.L. Breisch, An intuitive approach to speleotopology, Southwest Cavers 6 (1967) 72–78.
[3] M.R. Fellows, M.A. Langston, On search, decision, and the efficiency of polynomial-time algorithms, J. Comput. System Sci. 49 (3) (1994) 769–779.
[4] F.V. Fomin, P.A. Golovach, Graph searching and interval completion, SIAM J. Discrete Math. 13 (4) (2000) 454–464.
[5] F.V. Fomin, P. Heggernes, J.A. Telle, Graph searching, elimination trees, and a generalization of bandwidth, Algorithmica 41 (2) (2004) 73–87.
[6] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, NY, USA, 1979.
[7] L.M. Kirousis, C.H. Papadimitriou, Searching and pebbling, Theoret. Comput. Sci. 47 (2) (1986) 205–218.
[8] A.S. LaPaugh, Recontamination does not help to search a graph, J. ACM 40 (2) (1993) 224–245.
[9] T.D. Parsons, Pursuit-evasion in a graph, in: Theory and Applications of Graphs, in: Lecture Notes in Math., vol. 642, Springer-Verlag, 1978, pp. 426–441.
[10] S. Rao, A.W. Richa, New approximation techniques for some linear ordering problems, SIAM J. Comput. 34 (2) (2005) 388–404.
[11] N. Robertson, P.D. Seymour, Graph minors. I. Excluding a forest, J. Combin. Theory Ser. B 35 (1) (1983) 39–61.
[12] B. Yang, D. Dyer, B. Alspach, Sweeping graphs with large clique number, Discrete Math. 309 (18) (2009) 5770–5780.