Abstract

We examine the zero-visibility cops and robber graph searching model, which differs from the classical cops & robber game in one way: the robber is invisible. We show that this model is not monotonic. We also provide bounds on both the zero-visibility copnumber and monotonic zero-visibility copnumber in terms of the pathwidth.

1 Introduction

Using mobile agents to find and capture a mobile intruder is a well-studied graph theory problem. Depending on the restrictions placed on the agents and the intruder, the resulting pursuit can vary wildly. One common restriction placed on both the agents and the intruder is a speed limit; in some versions of this game, while the agents may only move along edges one at a time, the intruder may move from any position on the graph to any other along a path that does not contain any agents. In other versions, the agents may "jump" from a vertex to any other vertex. In still other games, one or both of the agents and the intruder have limited information about the other party's position; that is, one party or the other may only see the opposition if they are near one another, or alternatively, may never see each other until they stumble upon each other at the same vertex.

The cops and robber model was introduced independently by Winkler and Nowakowski [21] and Quilliot [23]. In this model, a slow, visible intruder (the robber) moves from vertex to adjacent vertex in a graph, while pursued by one or more slow, visible agents (the cops), who also move from vertices to adjacent vertices. In these first papers, copwin graphs were characterised; that is, graphs where exactly one cop is sufficient to capture.

Many questions have grown out of these papers. Recently, a characterisation of the $k$-copwin graphs has been discovered [7]; that is, graphs where $k$ cops are required. Bounds on the copnumber (the minimum number of cops needed to guarantee capture of a robber) of planar graphs [2] have been found, as well as other surfaces [6, 25]. Surveys of graph searching problems, including the cops and robber model, are available [3, 5, 11].

A variation of this problem dates back to Tošić in 1985 [27]. This corresponds to the cops and robber model with one exception: the robber is invisible. Certainly, this makes the problem more difficult for the cops, and intuitively seems to be more like the edge searching model introduced by Parsons [22]. This zero-visibility cops and robber model may also be taken as a particular instance of the $k$-visibility cops and robber problem, where both cops and robber move as in the standard Winkler-Nowakowski-Quilliot model, but the robber only becomes visible to the cops when he is at distance at most $k$ from some cop.

In these models, the analog of copnumber can be defined in different ways. Following [26] and [27], we define the zero-visibility copnumber of a graph $G$ to be the minimum number of cops needed to guarantee capture of an invisible robber in finite time. (Other authors [16, 17, 18] do not necessarily include the restriction to finite time, which works well with their application of the probabilistic method.)
The zero-visibility copnumber for paths, cycles, complete graphs and complete bipartite graphs were characterised in [27], as were graphs that are zero-visibility copwin. There are several constructions in [14] for graphs which require at most 2 cops to perform a zero-visibility search, but a characterisation remains open. An algorithm for determining the zero-visibility copnumber of a tree in quadratic time was given in [26], but the complexity of the problem for general graphs remains open. Most recent work on these topics has been on limited (but not zero) visibility [12, 13], and on the expected capture time of the robber in the zero or limited visibility case [1, 13, 16, 17, 18].

One topic that has been a mainstay of edge searching problems is monotonicity. Basically, a strategy is monotonic if, once a region has been guaranteed to be free of the robber, the cops may not move in such a way to allow the robber to re-enter that region. It is known that the edge search model is monotonic [4, 20], but that the connected edge search model is not [29]. In the original cops and robber game, it is hard to motivate a definition of monotonicity, as the robber was visible and slow. In the zero-visibility version this becomes a natural question again. We will show that more cops are needed to perform a monotonic search.

2 Preliminaries

We consider a pursuit-evasion game on a graph we refer to as zero-visibility cops and robber. The game is played on a simple connected graph $G$ between two opponents, referred to as the cop player and the robber player.

The cop controls the movements of a fixed number of cop pieces and the robber player controls the movement of a single robber piece; we refer to both players and their pieces as cops and robbers. The cop player begins by placing the cops on some set of vertices of $G$ and his opponent then places the robber on a vertex, unknown to the cop player. More than one cop piece may occupy a vertex. The players then alternate turns, beginning with the cop. On each player’s turn, he may move one or more of his pieces from its current vertex to an adjacent vertex; either player may leave any or even all of his pieces where they are. The game ends with a victory for the cop player if, at any point, the robber piece and a cop piece occupy the same vertex. The robber wins if this situation never occurs.

It is important to emphasise that, until he has won, the cop player has no information regarding the robber’s position or moves – he cannot see the robber piece until it and a cop occupy the same vertex. On the other hand, the cop may, due to his past moves, gain some knowledge on the possible locations of the robber. As well, note that the robber player is fully aware of all moves the cop makes.

2.1 Strategies and territory

All graphs are assumed to be simple; any two vertices are joined by at most one edge and there are no loops (edges from a vertex to itself). We introduce the following terminology regarding this game.

For a graph $G$, $V_G$ and $E_G$ are the vertex and edge sets of $G$. The relation $x \sim y$ holds if $x$ and $y$ are adjacent (joined by an edge). The relation $x \simeq y$ holds if $x \sim y$ or $x = y$. For each $X \subseteq V_G$, the set

$$\text{N}_G[X] = \{ x \in V_G \mid \exists y \in X \text{ such that } x \simeq y \}$$

is the closed neighbourhood of $X$. If $X = \{x\}$ is a singleton, we use $N_G[x]$ rather than $N_G[\{x\}]$ to represent the closed neighbourhood of $x$. For $X \subseteq V_G$, the boundary of $X$ is the set of vertices adjacent to members of $X$ but not contained in $X$:

$$\delta_G(X) = \{ y \notin X \mid \exists x \in X \text{ such that } x \sim y \} = N_G[X] \setminus X.$$

We will make extensive use of the concept of a walk in a graph; however, we give walks additional structure normally not present in their definition. We define a walk in a graph to be a (possibly infinite) sequence of vertices

$$\alpha = (\alpha(0), \alpha(1), \ldots)$$

such that for all $t \geq 0$, $\alpha(t+1) \simeq \alpha(t)$. We use walks to describe cops’ and robber’s movements; if a walk $\alpha$ corresponds to the positions of a single piece within the game – the vertex $\alpha(0)$ is the starting position of the piece and the vertex $\alpha(t)$ is the location of the piece after its controller has taken $t$ turns.
A strategy on \( G \) for \( k \) cops is a finite set of walks \( \mathcal{O} = \{l_i\}_{i=1}^k \), all of the same length \( T \) (possibly \( T = \infty \)).

A strategy \( \mathcal{O} \) corresponds to a potential sequence of moves by the cop player; each walk \( l_i \in \mathcal{O} \) corresponds to the moves of one of the cop pieces. The order of a strategy is the number of cop pieces required to execute it. If a strategy has length \( T < \infty \), we might imagine that the cop player forfeits if he hasn’t won after \( T \) moves.

We say that a strategy is successful if it results in a win for the cop player regardless of the moves made by the robber. Evidently, a strategy \( \mathcal{O} = \{l_i\}_{i=1}^k \) of length \( T \) is successful if and only if for every walk \( \alpha \) of length \( T \) in \( G \), there are \( l_i \in \mathcal{O} \) and \( t < T \) such that \( \alpha(t) = l_i(t) \) or \( \alpha(t) = l_i(t+1) \). The robber must be caught at some point, either by moving onto a cop or having a cop move onto it.

The zero-visibility copnumber of a connected graph \( G \) is the minimum order \( c_0 = c_0(G) \) among successful strategies on \( G \); it is the smallest number of cops required to guarantee capture of the robber. Clearly, a strategy that begins with a cop on every vertex is successful, so the zero-visibility copnumber is well-defined.

Typically, in pursuit-evasion games of this sort, only finite strategies are considered successful; the robber must be caught in a bounded number of turns. However, the following theorem shows that if we allow infinite strategies in the zero-visibility cops and robber game, any successful strategy (as defined above) will succeed in a bounded amount of time, whether or not the strategy itself is finite.

**Theorem 1** Let \( G \) be a graph. Every infinite successful strategy on \( G \) may be truncated to obtain a finite successful strategy.

**Proof:** Let \( \mathcal{O} = \{l_i\}_{i=1}^k \) be an infinite successful strategy on \( G \). We are concerned with showing that the robber that follows every (infinite) walk \( \alpha \) is caught within a bounded amount of turns.

For each infinite walk \( \alpha \) on \( G \) let

\[
\mathcal{r}(\alpha) = \inf \{ t \geq 0 \mid \exists i \text{ such that } \alpha(t) = l_i(t) \text{ or } \alpha(t) = l_i(t + 1) \}
\]

be the number of moves the robber makes before being caught, if the robber is following the walk \( \alpha \). Since \( \mathcal{O} \) is successful, every \( \mathcal{r}(\alpha) \) is finite. For each vertex \( x \) and for each nonnegative integer \( t \), let

\[
s(x, t) = \sup \left\{ \mathcal{r}(\alpha) \mid \alpha \text{ is a walk with } \alpha(t) = x \right\}.
\]

The (possibly infinite) value \( s(x, t) \) is the largest number of moves the robber can make before being caught, given that the robber is on vertex \( x \) after \( t \) moves. For some pairs \( x \) and \( t \), we will have \( s(x, t) \leq t - 1 \); this simply means that it is impossible for the robber to visit \( x \) on his \( t \)-th move as he will be caught before he can do so if he tries.

We note that if \( \alpha \) is a walk and \( 0 \leq t \leq \mathcal{r}(\alpha) \), then \( s(\alpha(t), t) \geq \mathcal{r}(\alpha) \), since, by definition, \( \alpha \) is a walk that allows the robber to make at least \( \mathcal{r}(\alpha) \) moves and visits \( \alpha(t) \) after \( t \) moves.

We now produce a contradiction—suppose that for every nonnegative integer \( t \), there is an infinite walk \( \alpha \) with \( \mathcal{r}(\alpha) \geq t \).

We first claim that there must be some vertex \( x \) with \( s(x, 0) = \infty \). For every \( t \geq 0 \), there is \( \alpha \) with \( \mathcal{r}(\alpha) \geq t \). Such an \( \alpha \) then has \( s(\alpha(0), 0) \geq t \). So, for every \( t \geq 0 \), there is a vertex \( x \) with \( s(x, 0) \geq t \); there are only finitely many vertices, so at least one \( x \) has \( s(x, 0) = \infty \).

We next claim that if \( t' \geq 0 \) and the vertex \( x \) has \( s(x, t') = \infty \), then there is a vertex \( y \simeq x \) that has \( s(y, t' + 1) = \infty \). Let \( t \geq t' + 1 \); since \( s(x, t') = \infty \), there is an infinite walk \( \alpha \) with \( \mathcal{r}(\alpha) \geq t \) and \( \alpha(t') = x \). We then have \( s(\alpha(t' + 1), t' + 1) \geq t \) and \( \alpha(t' + 1) \simeq \alpha(t') = x \), for any such \( \alpha \). Thus, if \( s(x, t') = \infty \), then for \( t \geq t' + 1 \), there is \( y \simeq x \) such that \( s(y, t' + 1) \geq t \). Again, there are only finitely many vertices adjacent to \( x \), so if \( s(x, t') = \infty \), there is \( y \simeq x \) such that \( s(y, t' + 1) = \infty \).

We note that if \( s(x, t) = \infty \), then for each \( i \in \{1, \ldots, k\} \), \( x \neq l_i(t) \) and \( x \neq l_i(t + 1) \). (If \( x = l_i(t) \) or \( x = l_i(t + 1) \) for some \( i \) then any \( \alpha \) with \( \alpha(t) = x \) has \( \mathcal{r}(\alpha) \leq t \).)

From the above, it is straightforward to construct a walk which allows the robber to evade the cops indefinitely, contradicting our assumption that \( \mathcal{O} \) is successful. The robber starts at a vertex \( x_0 \) with \( s(x_0, 0) = \infty \). Suppose that after \( t \) moves, the robber is on \( x \) with \( s(x, t) = \infty \); on his \( (t + 1) \)-th move he picks \( y \simeq x \) with \( s(y, t + 1) = \infty \). Following this strategy, the robber never occupies the same vertex as a cop.

3
So, if $O = \{l_i\}_{i=1}^k$ is an infinite successful strategy, then there is $t_O \geq 0$ such that for every infinite walk $\alpha$, $r(\alpha) \leq t_O$. Then, the strategy $O'$ obtained by replacing each

$$l_i = (l_i(0), l_i(1), \ldots) \text{ with } l'_i = (l_i(0), l_i(1), \ldots, l_i(t_O + 1))$$

is a finite successful strategy.

In light of Theorem 1, we will only consider finite strategies. Moreover, we can recast this game as a node-search style problem. Rather than imagine an opponent, we simply keep track of all possible vertices on which the robber piece might be found, via the following construction:

1. Initially, every vertex is marked as *dirty*.
2. A dirty vertex is *cleaned* if a cop piece occupies it.
3. In between each of the cop’s turns, every cleaned vertex that is unoccupied and adjacent to a dirty vertex becomes dirty.

The dirty vertices are the set of all possible locations of the robber. We refer to the step in between the cop’s turns where unoccupied cleaned vertices may become dirty as *recontamination*.

Let $G$ be a graph and let $O$ be a strategy of length $T$. For each nonnegative integer $t \leq T$,

1. let $O_t$ be the set of vertices occupied by cops at the end of the $t$-th turn by the cops;
2. let $R_t$ be the set of vertices that are dirty immediately before the cop’s $t$-th turn; and
3. let $A_t$ be the set of vertices that are dirty immediately after the cop’s $t$-th turn.

In other words, at the beginning of a $t$-th turn, $t \geq 1$, the cops occupy the vertices in $O_{t-1}$ and $R_t$ are the dirty vertices (possible locations of the robber). Then, the cops move and $O_t$ becomes the vertex set they occupy, and $A_t$ becomes the set of vertices that are dirty. After the following robber’s move, $R_{t+1}$ is the set of dirty vertices.

We define, somewhat arbitrarily, $R_0 = V_G$. For $t \geq 0$, the relevant rules of the game imply that

$$A_t = R_t \setminus O_t,$$  \hspace{1cm} (1)

$$R_{t+1} = N_G[A_t] \setminus O_t, \text{ and}$$  \hspace{1cm} (2)

$$O_{t+1} \subseteq N_G[O_t].$$  \hspace{1cm} (3)

A strategy of finite length $T$ is successful if and only if $A_T$ is empty. This labeling of the cop and robber territories is introduced in [26].

### 2.2 Weak monotonicity and vibrations

In a pursuit-evasion game of this sort, a topic of general interest is that of the monotonicity of strategies. Typically, a strategy is *monotonic* if recontamination never occurs. In this case, such a strategy would have

$$R_0 \supseteq A_0 \supseteq R_1 \supseteq A_1 \supseteq \ldots \supseteq R_T \supseteq A_T,$$

where $T$ is the length of the strategy.

However, consider the following possible activity of a single cop piece. Let $xy$ be an edge and suppose we are attempting to construct a strategy that cleans a graph $G$. If a single cop moves back and forth between $x$ and $y$, the two vertices $x$ and $y$ are guarded from the robber. If the robber moves onto either while this is occurring he will be caught either immediately or on the next turn.

Considering this activity under the graph cleaning model, although the vertices $x$ and $y$ are possibly being recontaminated over and over, the contamination can never “spread” through them, as they are cleaned before they can possibly recontaminate any further vertices.

We will refer to the above activity as *vibrating* on the edge $xy$. Further, if $E$ is a set of edges, we say that a set of cops is vibrating on $E$ if each is vibrating on a member of $E$ and every member of $E$ is thus
protected. We will also occasionally refer to a set of cops vibrating on a set of vertices $X$; this simply means that $X$ is covered by some set of edges $E$ and the cops are vibrating on $E$. Typically, we want a set of cops to vibrate on a matching (a set of edges that do not share any endpoints), in order to most efficiently utilise this tool.

We wish to take advantage of the above strategic element while still exploring the topic of monotonicity. Thus, we define a strategy of length $T$ to be weakly monotonic if for all $t \leq T-1$, we have $A_{t+1} \subseteq A_t$. In a weakly monotonic strategy, every time a cleaned vertex is recontaminated, it is cleaned on the very next move by the cop player. Thus, if the cop player is following a weakly monotonic strategy and the robber moves onto a vertex that has been previously visited by a cop, he will be caught on the very next move by the cop player. Thus, if the cop player is following a weakly monotonic strategy, we define a strategy of length $T$ to be weakly monotonic if for all $t \leq T-1$, we have $A_{t+1} \subseteq A_t$. In a weakly monotonic strategy, every time a cleaned vertex is recontaminated, it is cleaned on the very next move by the cop player. Thus, if the cop player is following a weakly monotonic strategy and the robber moves onto a vertex that has been previously visited by a cop, he will be caught on the very next turn.

The monotonic zero-visibility copnumber of a connected graph $G$ is the minimum order $mc_0 = mc_0(G)$ among successful weakly monotonic strategies on $G$; it is the smallest number of cops required to capture the robber utilising a weakly monotonic strategy. We are exclusively interested in weakly monotonic strategies as opposed to the stronger variant, and so we will simply use the term monotonic, with the understanding that this means weakly monotonic as defined above.

Clearly, we have $c_0(G) \leq mc_0(G)$ for all graphs $G$.

2.3 Cliques and matchings

A clique in a graph is a set of vertices that are all adjacent to each other. The clique number of the graph $G$, denoted as $\omega(G)$, is the maximum size of a clique in $G$. A complete graph is a graph whose vertex set is a clique: $K_n$ is the complete graph on $n$ vertices.

A matching in a graph is a set of edges such that no two are incident (share an endpoint). A matching $M$ covers a vertex $v$ if there is an edge in $M$ that has $v$ as an endpoint. The matching number of the graph $G$, $\nu(G)$, is the maximum size of a matching in the graph $G$. It is known that a maximum matching, and thus the matching number, can be found in polynomial time; see [9].

Theorems 2 and 3 provide bounds for the zero-visibility copnumbers utilising the well-understood graph parameters $\omega$ and $\nu$. Theorem 2 appears in [26, 27].

**Theorem 2** If $G$ is a connected graph, then $c_0(G) \geq \frac{1}{2}\omega(G)$. Moreover, $c_0(K_n) = mc_0(K_n) = \lceil \frac{n}{2} \rceil$.

**Theorem 3** If $G$ is a connected graph, then $mc_0(G) \leq \nu(G) + 1$.

**Proof:** Let $M$ be a matching in $G$ containing $\nu(G)$ edges. If a vertex $v$ is not covered by $M$, then every neighbour of $v$ must be covered by $M$, otherwise we could construct a strictly larger matching in $G$.

We present a monotonic strategy that cleans $G$ using either $\nu(G)$ or $\nu(G) + 1$ cops. We place $\nu$ cops, each on an endpoint of a distinct member of $M$. If there are any vertices not covered by $M$, we place one further cop on one of these. The cops on $M$ then begin vibrating on $M$.

If $M$ covers $G$, the cops win in the second round. If not, the cop that did not begin on $M$ visits the remainder of the vertices one vertex at a time, after which $G$ is cleaned. As noted above, there are no edges with both endpoints not covered by $M$, so the vibration on $M$ prevents these vertices from recontaminating each other. In either case, the strategy is monotonic.

2.4 The pathwidth of a graph

Let $G$ be a graph with vertex set $V_G$. A path decomposition of $G$ is a finite sequence $\mathcal{B} = (B_1, B_2, \ldots, B_n)$ of sets $B_i \subseteq V_G$ such that

1. $\bigcup_{i=1}^{n} B_i = V_G$;
2. if $x \sim y$, then there is $i \in \{1, \ldots, n\}$ such that $\{x, y\} \subseteq B_i$; and
3. if $1 \leq i < j < k \leq n$, then $B_i \cap B_k \subseteq B_j$. 

We refer to the sets $B_i$ as bags. An alternate, but equivalent, formulation of the third requirement is that for each vertex $x$, the bags that contain $x$ form a consecutive subsequence, $(B_i, B_{i+1}, \ldots, B_j)$, for some $i$ and $j$ with $1 \leq i \leq j \leq n$.

Let $G$ be a graph and let $B = (B_i)$ be a path decomposition of $G$. We define the width of $B$ to be one less than the maximum size of a bag,

$$pw(B) = \max \left\{ |B_i| - 1 \mid i \in \{1, \ldots, n\} \right\},$$

and the pathwidth of $G$ to be the minimum width of a path decomposition of $G$,

$$pw(G) = \min \{pw(B) \mid B \text{ is a path decomposition of } G\}.$$ 

The pathwidth of a graph has been introduced in [24].

**Lemma 1** Let $G$ be a connected graph with $pw(G) \leq |V_G| - 2$. Then, there is a path decomposition $B$ of $G$ containing $n \geq 2$ bags such that $pw(B) = pw(G)$ and, for each $i = 1, \ldots, n - 1$, each of

$$B_i \setminus B_{i+1}, \ B_{i+1} \setminus B_i \text{ and } B_i \cap B_{i+1}$$

is nonempty.

**Proof:** Let $B = (B_1, \ldots, B_n)$ be a path decomposition of $G$ such that $pw(B) = pw(G)$ and such that $n$ is minimal. By assumption $|V_G| \geq 2$, so $G$ contains at least one edge. As well, since $|B_i| - 1 \leq |V_G| - 2$ for all $i$, we must have $n \geq 2$.

If $B_i \cap B_{i+1}$ is empty for some $i \in \{1, \ldots, n - 1\}$, then there are no edges joining

$$B_1 \cup \ldots \cup B_i \text{ and } B_{i+1} \cup \ldots \cup B_n,$$

contradicting our assumption that $G$ is connected. So, we obtain that $B_i \cap B_{i+1}$ is nonempty for each $i = 1, \ldots, n - 1$.

Suppose that we delete one or more bags $B_i$, preserving the order of the remaining bags, to obtain a new sequence $B'$. For any vertex $x$, the sets in $B'$ that contain $x$ (if there are any) will still form a consecutive subsequence. So, the only ways in which $B'$ can fail to be a path decomposition is if for some vertex $x$ we have deleted every bag containing $x$ or if for some edge $xy$ we have deleted every bag containing both $x$ and $y$.

If there would be any consecutive pairs $B_i$ and $B_{i+1}$ where one of the two bags is contained in the other, we could delete the smaller of the two bags in such a pair (either one, but not both, if they are equal). If $B_i \subseteq B_{i+1}$ and we delete $B_i$, we cannot have deleted every bag containing a vertex or both endpoints of an edge. The case $B_{i+1} \subseteq B_i$ is the same. Hence, due to the nonminimality of $n$, $B_i \setminus B_{i+1}$ and $B_{i+1} \setminus B_i$ are both nonempty for all $1 \leq i \leq n - 1$. 

The pathwidth of a graph can be characterised via a pursuit-evasion game on a graph. Rather than describing the cop and robber dynamics of the game, we will simply define it as a model in cleaning a graph. In this game, the cops do not move along the edges of the graph. Each cop has two moves available to it (although at any point in time only one is possible):

1. if a cop is currently on a vertex in the graph, it may be “lifted” off the graph; and
2. if a cop is currently not in the graph, it may be “placed” on any vertex in the graph.

On each of the cop player’s turns, each of his pieces may make only one move. Thus, moving a cop from one vertex to another requires two turns. Initially, every edge is marked as dirty. An edge is cleaned when both of its endpoints are occupied by a cop. After each move by the cop player, a cleaned edge $e$ is recontaminated if there is a path that contains no cops and connects $e$ to a dirty edge – in this game, the contamination of the graph moves arbitrarily fast.

This pursuit-evasion game is often referred to as node-searching; it can be shown that $pw(G) \leq k$ if and only if there is a successful node-search strategy on $G$ utilising $k + 1$ cops [10, 19].

We introduced this second pursuit-evasion game as it is utilised in the proof of Lemma 5; the remainder of this work deals exclusively with the zero-visibility game previously defined.
3 Pathwidth and the zero-visibility copnumber

We prove a series of inequalities relating the zero-visibility copnumber, the monotonic zero-visibility copnumber and the pathwidth of a graph. In [28], a pursuit-evasion game referred to as strong mixed search is introduced. It is possible to prove the results in this section utilising the relationships shown therein between strong mixed search and the pathwidth of a graph.

A caterpillar is a tree such that deleting all vertices of degree 1 results in a path or an empty graph. The proof of Theorem 4 is a straightforward exercise and is omitted.

**Theorem 4** Let $G$ be a graph. The following are equivalent:

1. We have $c_0(G) = 1$, $mc_0(G) = 1$ or $pw(G) = 1$.
2. We have $c_0(G) = mc_0(G) = pw(G) = 1$.
3. The graph $G$ is a caterpillar.

In [26], it is shown that for any connected graph $G$, $c_0(G) \leq pw(G) + 1$. We improve this result slightly with Theorem 5. The treewidth of a graph $G$, $tw(G)$ is a parameter in a similar vein to the pathwidth. Theorem 5 is analogous to Proposition 1.5 in [15], where it is shown that the (visible) copnumber of a graph $G$ is less than or equal to $\frac{1}{2}tw(G) + 1$.

**Theorem 5** Let $G$ be a connected graph containing two or more vertices. Then, $c_0(G) \leq pw(G)$.

**Remark 1** The assumption that $G$ contains at least two vertices is necessary. The graph $G$ on a single vertex is the unique connected graph with $pw(G) < c_0(G)$.

**Proof:**

If $pw(G) = |V_G| - 1$, we simply begin with a cop on every vertex except for one, say $x$; we then move a cop from one of its neighbours onto $x$ and we have cleaned $G$ using $pw(G)$ cops. So, we assume that $pw(G) \leq |V_G| - 2$.

Let $B = (B_1, \ldots, B_n)$ be a path decomposition of $G$ that satisfies the conditions of Lemma 1 and let $k = pw(G) = pw(B)$. For $i = 1, \ldots, n - 1$, let

$$C_i = (B_1 \cup \cdots \cup B_i) \setminus B_{i+1}$$

be the set of vertices that appear in the first $i$ bags but not in the final $n - i$. Each such $C_i$ is nonempty since $B_i \setminus B_{i+1} \subseteq C_i$ is nonempty. We note that if $i \leq n - 1$,

$$C_{i+1} = C_i \cup (B_{i+1} \setminus B_{i+2}) \subseteq C_i \cup B_{i+1}.$$  

Recall that $\delta_G(X) = N_G[X] \setminus X$. We claim that $\delta_G(C_i) \subseteq B_i \cap B_{i+1}$. Suppose that $x \in C_i$, $y \notin C_i$ and $x \sim y$. Since

$$x \notin B_{i+1} \cup \cdots \cup B_n,$$

the bag that contains both $x$ and $y$ must be one of the first $i$ bags – say $B_i$, where $i' \leq i$. However, $y \notin C_i$ and

$$y \in B_{i'} \subseteq B_1 \cup \cdots \cup B_i$$

implies that $y \in B_{i+1}$. Thus, $y \in B_i \cap B_{i+1} \subseteq B_i$, further implying that $y \in B_i \cap B_{i+1}$.

We present a strategy that cleans $G$ and utilises $k$ cops. Since $B_i \setminus B_{i+1} \neq \emptyset$ and $B_i \cap B_{i+1} \neq \emptyset$, for each $i \in \{1, \ldots, n - 1\}$,

$$|B_i \cap B_{i+1}| \leq k.$$  

Our initial placement and first move are very simple. If $|B_1| \leq k$, we simply begin with a cop on every vertex in $B_1$; any other cops can be placed arbitrarily. Otherwise, we pick an edge $xy$ with $x \in B_1 \setminus B_2$ and $y \in B_1 \cap B_2$; such an edge exists because $C_1 = B_1 \setminus B_2 \neq V_G$ is nonempty and $\delta_G(C_1) \subseteq B_1 \cap B_2$. We
begin with a cop on every member of $B_1$ except for $y$; on the cop player’s first turn, we move the cop from $x$ to $y$. After this move, each vertex in $C_1$ is cleaned and there is a cop on every vertex in $B_1 \cap B_2$.

Now, assume that, after some number of turns, each vertex in $C_i$ is cleaned and there is a cop on every vertex in $B_i \cap B_{i+1}$. This layout of cop pieces prevents any recontamination of $C_i$, because $\delta_G(C_i) \subseteq B_i \cap B_{i+1}$.

We then move cops to as many distinct vertices in $B_{i+1} \setminus B_i$ as possible, without removing any of the cops from $B_i \cap B_{i+1}$. This will either result in a cop on every vertex in $B_{i+1}$ or a cop on all vertices except for one vertex $y \in B_{i+1} \setminus B_i$.

First, suppose that $i = n - 1$. If there are no dirty vertices left (if $|B_n| \leq k$), we are done. Otherwise, if there is a dirty vertex $y \in B_n \setminus B_{n-1}$, its neighbours are all occupied by cops (it cannot have a neighbour in $C_{n-1}$ because $\delta_G(C_{n-1}) \subseteq B_{n-1} \cap B_n$). One of these cops moves from a neighbouring vertex to $y$ and the graph is now cleaned.

So, suppose that $i \leq n - 2$. If $|B_{i+1}| \leq k$, there is now a cop on every vertex in $B_{i+1}$. So, we have cleaned

$$C_{i+1} = C_i \cup (B_{i+1} \setminus B_{i+2}) \subseteq C_i \cup B_{i+1}$$

and there is a cop on every vertex in $B_{i+1} \cap B_{i+2}$; we can then reiterate this process.

Otherwise, if $B_{i+1}$ contains $k + 1$ vertices, we have $y \in B_{i+1} \setminus B_i$ unoccupied by a cop (and not yet cleaned).

If $y \in B_{i+1} \setminus B_{i+2}$, then $y \in B_{i+1}$ but $y \notin B_i \cup B_{i+2}$. Any neighbour of $y$ is then a member of $B_{i+1}$, all of which are occupied by cops. So, one cop on a neighbour $x$ of $y$ moves onto $y$ and then back to $x$ (while all other cops remain where they are). After these two moves, each vertex in $C_{i+1}$ is cleaned and every vertex in $B_{i+1} \cap B_{i+2}$ contains a cop, as the only vertex in $B_{i+1}$ which does not contain a cop is $y \notin B_{i+2}$.

If $y \in B_{i+1} \cap B_{i+2}$, we note that at this point every vertex in $C_{i+1} = C_i \cup B_{i+1} \setminus B_{i+2}$ has been cleaned and the only vertex in $\delta_G(C_{i+1})$ that does not contain a cop is $y$. We then pick a vertex $x \in B_{i+1} \setminus B_{i+2}$; the vertex $x$ is occupied by a cop. We move the cop from $x$ to $y$, in one or more moves. The only possible dirty neighbour of $x$ is $y$, in which case the cop moves directly onto $y$; so, in any event, we can move the cop from $x$ to $y$ without allowing $x$ to be recontaminated. Once this is complete, each vertex in $C_{i+1}$ is cleaned and every vertex in $B_{i+1} \cap B_{i+2}$ contains a cop.

After either of the above two possibilities, we can then reiterate this process. □ □

**Theorem 6** Let $G$ be a connected graph; then, $mc_0(G) \leq 2 \text{pw}(G) + 1$.

**Proof:** We refer to a path decomposition $B = (B_1, \ldots, B_n)$ as *connected* if, for each $i \in \{1, \ldots, n\}$, the subgraph induced by $B_1 \cup \cdots \cup B_i$ is connected. For a graph $G$, the *connected pathwidth*, denoted by $\text{cpw}(G)$, is the minimum width of a connected path decomposition of $G$. In [8] it is shown that for a connected graph $G$, $\text{cpw}(G) \leq 2 \text{pw}(G) + 1$.

Suppose that $B$ is a connected path decomposition of $G$ of width $k = \text{cpw}(G) \leq 2 \text{pw}(G) + 1$. It can be shown that deleting bags in $B$ until the conditions of Lemma 1 are met will not alter the fact that $B$ is connected. So, we can construct a strategy on $G$ utilising $k$ cops in the same manner as in Theorem 5. Moreover, since each

$$B_1 \cup \cdots \cup B_i = C_{i-1} \cup B_i$$

is connected, a simple modification of the strategy allows us to proceed in a monotonic fashion utilising $k$ cops.

For $i \in \{1, \ldots, n\}$, let $G_i$ be the subgraph induced by $B_1 \cup \cdots \cup B_i$. We begin with every cop placed on distinct vertex of $G_1$. Suppose that each vertex in $C_i$ is cleaned, there is a cop on every vertex in $B_i \cap B_{i+1}$ and suppose further that

1. every cop is on a distinct vertex of $G_i$, and
2. up until this point, every vertex that has become recontaminated has been cleaned on the very next turn.
When we move the remaining cops (those not occupying $B_i \cap B_{i+1}$) onto $B_{i+1} \setminus B_i$ (possibly leaving one vertex unoccupied), we do so in the following manner:

1. We move the cops one at a time.
2. Each cop moves from its starting point (some vertex in $G_i$) to a vertex in $B_{i+1} \setminus B_i$ without leaving $G_{i+1}$; this is possible because $G_{i+1}$ is connected and contains $G_i$ as a subgraph.
3. Each cop stops on the first unoccupied vertex in $B_{i+1} \setminus B_i$; it moves onto.

This prevents any recontamination as we move the cops onto $B_{i+1} \setminus B_i$. The other cop movements in the strategy can similarly be controlled so that either no recontamination occurs or any recontaminated vertices are immediately cleaned.

Thus, connected path decompositions give rise to weakly monotonic strategies and we have $mc_w(G) \leq cpw(G) \leq 2pw(G) + 1$. □ □

**Theorem 7** Let $G$ be a connected graph; then, $pw(G) \leq 2mc_w(G) - 1$.

**Proof:** Let $O$ be a successful monotonic strategy on $G$ that utilises $k$ cops. Let $T$ be the length of $O$; if $T \leq 1$, then $k$ cops can clean $G$ either after their initial placement or after one turn. This implies that $G$ contains at most $2k$ vertices. Simply letting $B_1$ be the entire vertex set results in a path decomposition with width less than or equal to $2k - 1$. So, we assume that $T \geq 2$.

For $t \leq T$, let $O_t$ and $A_t$ be as defined previously. We have $A_0 = V_G \setminus O_0$ and, since $O$ is a monotonic strategy, $A_t \subseteq A_{t-1}$, for $2 \leq t \leq T$. We note that for $t \geq 1$, if a cop visits a vertex during the $t$th move, then $x \notin A_t$ and $x$ has not been visited by a cop during the first $t$ moves, then $x \in A_t$.

Thus, the ($t-1$)th turn can be controlled so that either no recontamination occurs or any recontaminated vertices are immediately cleaned.

Thus, connected path decompositions give rise to weakly monotonic strategies and we have $mc_w(G) \leq cpw(G) \leq 2pw(G) + 1$. □ □

Let $G$ be a connected graph; then, $pw(G) \leq 2mc_w(G) - 1$.

**Proof:** Let $O$ be a successful monotonic strategy on $G$ that utilises $k$ cops. Let $T$ be the length of $O$; if $T \leq 1$, then $k$ cops can clean $G$ either after their initial placement or after one turn. This implies that $G$ contains at most $2k$ vertices. Simply letting $B_1$ be the entire vertex set results in a path decomposition with width less than or equal to $2k - 1$. So, we assume that $T \geq 2$.

For $t \leq T$, let $O_t$ and $A_t$ be as defined previously. We have $A_0 = V_G \setminus O_0$ and, since $O$ is a monotonic strategy, $A_t \subseteq A_{t-1}$, for $2 \leq t \leq T$. We note that for $t \geq 1$, if a cop visits a vertex during the $t$th move, then $x \notin A_t$ and $x$ has not been visited by a cop during the first $t$ moves, then $x \in A_t$.

Thus, connected path decompositions give rise to weakly monotonic strategies and we have $mc_w(G) \leq cpw(G) \leq 2pw(G) + 1$. □ □
Let Corollary 1 after which both $x$ and $y$ have been visited by a cop. Thus, we note that $y$ is the first time a cop visited $x$. This implies that the bags which contain $x$ form a consecutive subsequence of $\mathcal{B}$.

For $t = 2$, the same reasoning as above proves the claim; we simply replace all instances of $A_t-2$ in the above discussion with $V_G$. We now show that the sequence $\mathcal{B}$ forms a path decomposition.

Let $x \in V_G$, and suppose that $x \in B_t \setminus B_{t-1}$. By 4, $x \in A_{t-1} \setminus A_t$, further implying that $x \in A_s$ for all $s \leq t - 1$. Since $B_s = N_G[A_{s-1}] \setminus A_s$, we see that $x \notin B_s$ for all $s \leq t - 1$. So, if $x \notin B_{t-1}$ and $x \in B_t$, then $B_t$ is the first bag that contains $x$. This implies that the bags which contain $x$ form a consecutive subsequence of $\mathcal{B}$.

Now, suppose that $x \sim y$. Let $t$ be the first time a cop visited $x$ and let $s$ be the first time a cop visited $y$. If $0 \leq s, t \leq 1$, then $x, y \in B_1$. If $2 \leq t = s \leq T$, we have $x, y \in A_{t-1} \setminus A_t \subseteq B_t$. If $2 \leq t < s \leq T$, then we note that $x \notin A_t$ and $s \notin A_s$ via monotonicity. However, $y \in A_{s-1}$ and $x \sim y$, so $x \in N_G[A_{s-1}]$. Thus, $x \in N_G[A_{s-1}] \setminus A_s = B_s$. As well, $y \in A_{s-1}$ and $y \notin A_s$ imply that $y \in A_{s-1} \setminus A_s \subseteq B_s$. We conclude that if $x \sim y$, then $x$ and $y$ are both contained in $B_t$, where $s$ is the smallest number of moves after which both $x$ and $y$ have been visited by a cop. \hfill $\Box$ \hfill $\Box$

**Corollary 1** Let $G$ be a connected graph on two or more vertices. Then,

$$c_0(G) \leq pw(G) \leq 2mc_0(G) - 1 \leq 4pw(G) + 1.$$  

In Section 4 we provide constructions that in particular prove that the bound in Theorem 5 is tight and the bound in Theorem 6 is tight up to a small additive constant. Moreover, we also use Theorem 2 to argue that the bound in Theorem 7 is tight as well.

### 4 Comparing the zero-visibility copnumbers and the pathwidth of a graph

We examine a series of constructions that illuminate the relationships between the pathwidth and the zero-visibility copnumbers of a given graph. We show that the inequalities in Corollary 1,

$$c_0(G) \leq pw(G) \leq 2mc_0(G) - 1 \leq 4pw(G) + 1,$$

are either the best possible, or very close to being so.

By Theorem 4, the class of graphs with zero-visibility copnumber equal to 1 is identical to the class of graphs with pathwidth equal to 1. However, if $c_0(G) = 2$, this gives us absolutely no information concerning $mc_0(G)$ or $pw(G)$, as we will see in Theorem 8.

#### 4.1 Subdivisions of binary trees

Let $G$ be a connected graph; the distance between any two vertices $x$ and $y$ is the minimum length of a path connecting $x$ and $y$ and is denoted $d_G(x, y)$. So, if $d_G(x, y) = k \geq 1$, then there is a path connecting $x$ and $y$ of length $k$ and there are no shorter such paths. If $H$ is a connected subgraph of $G$, we have $d_H(x, y) \geq d_G(x, y)$ for all $x$ and $y$ in $V_H$. We say that $H$ is an isometric subgraph of $G$ if $d_H(x, y) = d_G(x, y)$ for all $x$ and $y$ in $V_H$.

**Lemma 2** Let $G$ be a graph. If $H$ is an isometric subgraph of $G$, then $c_0(H) \leq c_0(G)$.  

10
Let $G$ be a graph; we refer to an edge $e = xy$ as a cut edge if the graph $G - e$ obtained by deleting $e$ has more connected components than $G$ has. Clearly, if $G$ is connected and $e$ is a cut edge, then $G - e$ has two connected components. Not every graph contains cut edges.

**Lemma 3** Let $G$ be a connected graph that contains a cut edge $e$. If $H$ is one of the connected components of $G - e$, then

$$c_0(H) \leq c_0(G) \text{ and } mc_0(H) \leq mc_0(G).$$

Moreover, let $O$ be a successful strategy on $G$. Then, at some point in the strategy at least $c_0(H)$ cops are simultaneously present in $H$; if $O$ is a monotonic strategy, then at some point at least $mc_0(H)$ cops are simultaneously present in $H$.

**Proof:** Let $e = xy$ be a cut edge of $G$ and let $H$ be the connected component of $G - e$ that contains $x$. Let $k' \geq c_0(G)$, let $O = \{l_i\}_{i=1}^{k'}$ be a successful strategy on $G$ and let $O_0, \ldots, O_T$ be the cop positions mandated by $O$. We argue that there is a nonnegative $t \leq T$ such that

$$|O_t \cap V_H| \geq c_0(H)$$

and if $O$ is monotonic, then

$$|O_t \cap V_H| \geq mc_0(H).$$

We can then assume that $O$ utilises as few cops as possible, implying that for each $t$, $|O_t| \leq c_0(G)$ or $|O_t| \leq mc_0(G)$ (again, depending on whether $O$ is monotonic or not). This then implies each of the statements of the lemma. The same will be true for the connected component containing $y$, via a symmetric argument.

Let

$$k = \max_{0 \leq t \leq T} \{|O_t \cap V_H|\}$$

be the largest number of cops simultaneously present in $H$. We produce a successful strategy $O' = \{l'_i\}_{i=1}^{k}$ on $H$.

For each $i \in \{1, \ldots, k\}$ with $l_i(0) \in V_H$ we place a cop on $l_i(0)$ in $O'$. Then, we place

$$k - |\{i \in \{1, \ldots, k\} \mid l_i(0) \in V_H\}|$$

additional (if any) cops on $x$ (so that $O'$ places on the vertices of $H$ exactly $k$ cops in total) and mark these cops as “inactive".

We then follow the moves laid out by $O$, in the following manner. For each $t \in \{1, \ldots, T\}$ we have:

1. whenever a cop enters $H$ in $O$, that is, $l_i(t) = x$ and $l_i(t - 1) \notin V_H$ for some $i \in \{1, \ldots, k'\}$, then we select any “inactive” cop and we mark it as “active’. We refer to the latter as the cop corresponding to the cop $i$ until it becomes “inactive” again. (The activated cop begins mirroring the moves dictated by $O$ by following $l_i$ in $H$.)

2. whenever a cop $i, i \in \{1, \ldots, k'\}$, exits $H$ in $O$, that is $l_i(t - 1) = x$ and $l_i(t) \notin V_H$, then the corresponding cop remains on $x$ and it is marked as “inactive”;

3. whenever a cop $i, i \in \{1, \ldots, k'\}$, moves within $H$ in $O$, that is, $l_i(t) \in V_H$ and $l_i(t - 1) \in V_H$, then the corresponding cop, say $i' \in \{1, \ldots, k\}$, makes the same move in $O'$, i.e., $l'_i(t) = l_i(t)$; and

4. the remaining inactive cops on $x$ do not move.

Since $O$ is successful on $G$, the strategy described above will successfully clean $H$; if $O$ is monotonic, this new strategy will be so as well (every visit to a vertex in $V_H$, except for possibly the initial placement of cops on $x$, is a visit already mandated by $O$). So, we must have $k \geq c_0(H)$ or, if $O$ is monotonic, $k \geq mc_0(H)$. □

**Corollary 2** Let $T$ be a tree. If $T'$ is a subtree of $T$, then $mc_0(T') \leq mc_0(T)$. □
Figure 1: The subgraph obtained by deleting the two paths of length 8 has strictly higher monotonic zero-visibility copnumber than the supergraph.

**Proof:** We proceed in a very simple manner by induction on the number of vertices in \( T \). If \( |V_T| \leq 6 \), then the monotonic zero-visibility copnumber of \( T \) and any of its subtrees is 1 (any tree with 6 or fewer vertices is a caterpillar; see Theorem 4).

So, suppose that \( T \) is a tree on \( n \geq 7 \) vertices, \( T' \) is a proper subtree of \( T \) and that the proposition holds for trees on fewer than \( n \) vertices. Let \( e = xy \) be an edge in \( T \) where \( x \in V_{T'} \) and \( y \notin V_{T'} \); let \( T_1 \) be the connected component of \( T - e \) that contains \( x \). So, \( T' \) is a subtree of \( T_1 \).

By Lemma 3, we have \( mc_0(T_1) \leq mc_0(T) \), and via the inductive hypothesis, we have \( mc_0(T') \leq mc_0(T_1) \).

**Example 1** We produce an interesting example of a graph \( G \) with an isometric subgraph \( H \) such that \( mc_0(G) < mc_0(H) \). This illustrates that Lemmas 2 and 3 and Corollary 2 are limited in how they might be extended.

The graph \( G \) in question (see Figure 1) contains a large number of vertices with degree 2; to simplify the depiction, some of them are omitted. Specifically, the two dashed lines are paths of length 8; they each contain 7 internal degree-2 vertices which are not shown. The subgraph \( H \) is obtained by deleting the two paths drawn as dashed lines, it is straightforward to show that we cannot clean the subgraph monotonically with only two cops.

A **rooted tree** is a tree \( T \) where a single vertex has been marked as the **root**. In a rooted tree with root \( r \) every vertex \( x \neq r \) has a unique parent, identified in the following manner. Every vertex \( x \neq r \) is connected to \( r \) by a unique path; the parent of \( x \) is the sole neighbour of \( x \) in this path. If \( y \) is the parent of \( x \), then \( x \) is a child of \( y \); we also use the similarly defined terms grandparent and grandchild when discussing rooted trees. A **leaf** in a rooted tree is a vertex with no children. A rooted tree is **binary** if every vertex has at most two children. The **height** of a rooted tree is the maximum length of a path connecting the root and a leaf.

**Example 2** The following family of trees illustrates the distinction between the pathwidth of a tree and its monotonic zero-visibility copnumber. Let \( T_k \) be obtained by beginning with the full rooted binary tree of height \( k \) and subdividing every edge exactly once. We draw the root of each \( T_k \) with a circle; see Figure 2 for the first three such trees. In Lemmas 4 and 5, below, we will show that

\[
mc_0(T_k) = k \quad \text{and} \quad pw(T_k) = c_0(T_k) = \left\lfloor \frac{k}{2} \right\rfloor + 1.
\]
Algorithm 1 CLEAN($T_k$)

**Require:** There are $k + 1$ cops on the root of the graph $T_k$.

if $k = 0$ then
    return
end if

1. Let $x$ be the root of $T_k$ and let $y$ and $z$ be the two grandchildren of $x$. Let $T^y$ and $T^z$ be the two copies of $T_{k-1}$ rooted at $y$ and $z$, respectively.
2. In two moves, move $k$ of the cops from the root of $T_k$ to $y$, leaving the remaining cop on $x$.
3. CLEAN($T^y$).
4. Move all $k$ cops from $T^y$ to $z$ (each cop does not move any further into $T^z$).
5. CLEAN($T^z$).

return

Algorithm 1 cleans $T_k$ with a monotonic strategy that utilises $k + 1$ cops and begins with every cop placed on the root (this can be shown simply via induction). However, this is not an optimal strategy. If $k \geq 1$, there is a successful monotonic strategy on $T_k$ using $k$ cops. We obtain this strategy by using Algorithm 1 as follows. We first express $T_k$ as two copies of $T_{k-1}$ connected by a path of length 4:

$$T_k = \circ \\
T^1 = \bullet \circ \bullet \bullet \\
T^2 = \bullet \circ \bullet \bullet$$

We clean one copy of $T_{k-1}$ by using Algorithm 1 in such a way that one cop remains on the root of this subtree during the entire strategy. All $k$ cops then move to the root of the other copy of $T_{k-1}$ and clean that subtree in a similar fashion. Thus, $mc_0(T_k) \leq k$, if $k \geq 1$. In Lemma 4, we show that this strategy is, in fact, optimal.

**Lemma 4** For $k \geq 1$, $mc_0(T_k) = k$.

**Proof:** We prove, via induction on $k$, the following proposition: For $k \geq 1$, $mc_0(T_k) = k$; however, if a strategy begins with every cop placed on the root of $T_k$, $k + 1$ cops are required to clean the tree in a monotonic fashion.

For $k \in \{1, 2\}$, the truth of the statement is clear. So, suppose that $k \geq 2$ and $mc_0(T_k) = k$; suppose further that any successful monotonic strategy on $T_k$ beginning with every cop placed on the root utilises at least $k + 1$ cops.

We can express $T_{k+1}$ as two copies of $T_k$, labeled $T^{(1)}$ and $T^{(2)}$, with their roots connected via a path of length 4:

$$T_{k+1} = \bullet \circ \bullet \\
T^{(1)} = \circ \\
T^{(2)} = \bullet \circ \bullet$$

Now, suppose that we are attempting to clean $T_{k+1}$ in a monotonic manner utilising strictly fewer than $k + 1$ cops. By Lemma 3 and the inductive hypothesis, at some point in our strategy we must have $k$ cops simultaneously present in one of the subtrees $T^{(i)}$. Hence we must be using $k$ cops. Consider the
first time at which all \( k \) cops are present in one of the subtrees; we assume without loss of generality that this subtree is \( T^{(1)} \). At this point in time there are still dirty vertices in \( T^{(2)} \); since we are cleaning the tree monotonically, it must be that as of yet no cop has visited \( T^{(2)} \). In order to clean the entire graph, all \( k \) cops must, at some point, move into \( T^{(2)} \). However, this then induces a strategy on \( T^{(2)} \) in which every cop begins at the root – such a strategy requires at least \( k + 1 \) cops (by the inductive hypothesis). So, a strategy on \( T_{k+1} \) that utilises fewer than \( k + 1 \) cops cannot successfully clean the tree in a monotonic fashion. Our discussion involving Algorithm 1 then implies that \( mc_0(T_{k+1}) = k + 1 \).

Similarly, suppose that we are attempting to clean \( T_{k+1} \) in a monotonic manner beginning with every cop placed on the root. At some point, we must move at least \( k + 1 \) cops into one of the subtrees (since these cops are entering that subtree via its root). However, while we do so, we must leave an additional cop behind in order to prevent the root of \( T_{k+1} \) from being recontaminated. Thus, if we begin with every cop on the root of \( T_{k+1} \), we must utilise \( k + 2 \) cops to clean the tree in a monotonic fashion.

Theorem 7 and Lemma 4 together imply that \( pw(T_k) \leq 2k - 1 \). However, we in fact have \( pw(T_k) = \left\lceil \frac{k}{2} \right\rceil + 1 \).

Lemma 5 For \( k \geq 1 \), \( pw(T_k) = c_0(T_k) = \left\lceil \frac{k}{2} \right\rceil + 1 \).

Proof: In this proof we utilise the node-search game described in Section 2.4, rather than the zero-visibility cops & robber game. We show that \( T_k \) can be node-searched using \( \left\lfloor \frac{k}{2} \right\rfloor + 2 \) cops. Theorem 5 then implies that

\[
c_0(T_k) \leq pw(T_k) \leq \left\lfloor \frac{k}{2} \right\rfloor + 1.
\]

A proof which we omit, as it is very similar to that of Lemma 4, shows that \( c_0(T_k) = \left\lfloor \frac{k}{2} \right\rfloor + 1 \) and thus leads to the statement above.

Examination shows that \( T_1 \) and \( T_2 \) can be node-searched using 2 and 3 cops, respectively; moreover, the node-search on \( T_2 \) can be carried out by first placing a cop on the root and then leaving this first cop on the root throughout the entire game. We use induction on \( k \) to prove the following claim: If \( k \geq 1 \), then

1. \( T_k \) can be node-searched utilising \( \left\lfloor \frac{k}{2} \right\rfloor + 2 \) cops, and
2. if \( k \) is even, this search can be carried out with a stationary cop on the root.

Assume that \( k \geq 2 \) and that this claim holds for each \( k' \in \{1, \ldots, k\} \).

First, suppose that \( k \) is even and express \( T_{k+1} \) as two copies of \( T_k \):

\[
T_{k+1} = \begin{array}{c}
\bullet \\
T^{(1)} \\
\bullet
\end{array} \quad \begin{array}{c}
\bullet \\
\bullet \\
T^{(2)}
\end{array}
\]

Each \( T^{(i)} \), \( i = 1, 2 \), can be searched using \( \left\lfloor \frac{k}{2} \right\rfloor + 2 = \frac{k}{2} + 2 \geq 3 \) cops with one cop left stationary on the root. We perform such a search on \( T^{(1)} \) and then remove all cops except for the one on the root of \( T^{(1)} \). We then have this cop and one other “leapfrog” from the root of \( T^{(1)} \) to the root of \( T^{(2)} \). At this point every edge not in \( T^{(2)} \) is cleaned and we have a cop on the root of \( T^{(2)} \). We then clean \( T^{(2)} \), again, while leaving a cop on the root. Thus, \( T_{k+1} \) can be cleaned with \( \frac{k}{2} + 2 = \left\lfloor \frac{k+1}{2} \right\rfloor + 2 \) cops.

Next, suppose that \( k \) is odd and express \( T_{k+1} \) as four copies of \( T_{k-1} \) joined via the appropriate subdivided binary tree:

\[
T_{k+1} = \begin{array}{c}
\bullet \\
x_1 \\
\bullet
\end{array} \quad \begin{array}{c}
x_2 \\
x_3 \\
\bullet
\end{array} \quad \begin{array}{c}
x_4 \\
\bullet
\end{array} \quad \begin{array}{c}
\bullet \\
T^{(1)} \\
\bullet \\
T^{(2)} \\
\bullet \\
T^{(3)} \\
\bullet \\
T^{(4)}
\end{array}
\]
Each \( T^{(i)} \) is a copy of \( T_{k-1} \) and so can be searched using \( \left\lfloor \frac{k-1}{2} \right\rfloor + 2 = \frac{k-1}{2} + 2 \geq 3 \) cops while leaving one stationary on the root. We place a cop on the root of \( T_{k+1} \); this cop will not be moved throughout our entire strategy. Next, we perform a search on \( T^{(1)} \) with a stationary cop on the root and remove every cop except the one on the root of \( T^{(1)} \). We then perform the following sequence of moves:

1. place a cop on \( x_3 \),
2. remove the cop from the root of \( T^{(1)} \) and place it on \( x_2 \),
3. remove the cop from \( x_3 \) and place it on \( x_1 \),
4. remove the cop from \( x_1 \) and place it on \( x_4 \),
5. remove the cop from \( x_2 \) and place it on the root of \( T^{(2)} \), and then
6. remove the cop from \( x_4 \).

We can then search \( T^{(2)} \) with \( \left\lfloor \frac{k-1}{2} \right\rfloor + 2 \) cops, leaving a stationary cop on the root. The other branch of \( T_{k+1} \) is then cleaned in the same manner. So, \( T_{k+1} \) is cleaned using one stationary cop on the root and additional \( \left\lfloor \frac{k-1}{2} \right\rfloor + 2 \) cops; when \( k \) is odd, \( \frac{k-1}{2} + 3 = \left\lfloor \frac{k+1}{2} \right\rfloor + 2 \).

**Remark 2** The bound \( c_0(G) \leq pw(G) \) in Theorem 5 is the best possible.

By Lemmas 4 and 5, the subdivided binary trees \( T_k \), described in Example 2, satisfy
\[
c_0(T_k) = pw(T_k) < mc_0(T_k) \quad \text{for each } k > 2.
\]
Thus, the bound \( c_0(G) \leq pw(G) \) is tight on an infinite family of graphs. Moreover, we cannot bound pathwidth below by the monotonic variant of the zero-visibility copnumber.

**Remark 3** The bound \( mc_0(G) \leq 2pw(G) + 1 \) in Theorem 6 can only be improved by a small constant, if it can be improved at all.

The subdivided binary trees \( T_k \) satisfy
\[
mc_0(T_k) = k \quad \text{and} \quad pw(T_k) = \left\lfloor \frac{k}{2} \right\rfloor + 1.
\]
So,
\[
mc_0(T_k) = \begin{cases} 
2pw(T_k) - 1 & \text{if } k \text{ is odd,} \\
2pw(T_k) - 2 & \text{if } k \text{ is even.}
\end{cases}
\]
The coefficient of 2 cannot be reduced; only the additive constant can be changed, possibly by reducing it by one or two. In a roundabout manner, this shows that the result in [8] used in the proof of Theorem 6 is close to optimal.

### 4.2 Adding a universal vertex to a heavily subdivided tree

We produce a construction of graphs with \( c_0 < pw \) and \( c_0 < mc_0 \) with unbounded ratios in both inequalities.

A **universal vertex** in a graph \( G \) is a vertex adjacent to every other vertex. Given a graph \( G \), we form the graph \( G^\ast \) by adding a universal vertex to \( G \); a single new vertex is added, together with edges joining this new vertex and every other vertex already present in \( G \).

A **subdivision** of a graph \( G \) is a graph \( H \) formed by replacing one or more edges in \( G \) with paths of length greater than or equal to 2. If the vertices of \( G \) are labeled and \( H \) is a subdivision of \( G \), we preserve the labeling of the vertices, adding new labels to the new vertices.

**Lemma 6** If \( G \) is a tree containing two or more vertices, then there is a subdivision \( H \) of \( G \) such that \( c_0(H^\ast) = 2 \).
Proof: We proceed via strong induction on trees, with $G_1 \leq G_2$ if $G_1$ is a subtree of $G_2$. We prove the following, stronger statement: Let $G$ be a rooted tree with root $r$. Then, there is a subdivision $H$ of $G$ and a successful zero-visibility strategy on $H^*$ utilising two cops such that

(i) a cop visits the universal vertex $u$ at least every second turn throughout the game; and

(ii) once the root $r$ has been visited by a cop for the first time, either the game is finished or this cop vibrates on the edge $ru$ for the remainder of the game.

The base case is that $G$ is a path. In this case, we do not need to form a subdivision and hence we take $H = G$. A single cop vibrates on the edge joining the root of $G$ and the universal vertex of $H^*$; the subgraph induced by the remaining vertices forms either one or two paths, depending on whether the root is an endpoint or an internal vertex of $G$. The second cop then cleans these one or two paths from end to end. (In the extreme case that $G$ is a path of length 0, i.e., a single vertex, we still use this strategy, even though the second cop is unnecessary.)

Now, suppose that $G$ is a rooted tree and that the conditions (i) and (ii) hold for all rooted subtrees of $G$ (not equal to $G$).

Let $s$ be a child of $r$ and let $G_2$ be the subtree of $G$ consisting of $s$ and its descendants. Let $G_1$ be the subtree of $G$ consisting of all vertices in $V_G \setminus V_{G_2}$. The vertices $r$ and $s$ are selected as the roots of $G_1$ and $G_2$, respectively; see Figure 3. The vertex $r$ may have degree 1, in which case $G_1$ is a path of length 0.

Let $H_1$ and $H_2$ be subdivisions of $G_1$ and $G_2$ such that $H_1^*$ and $H_2^*$ can be cleaned using two cops subject to conditions (i) and (ii). Let $T$ be the number of turns (by the cop player) required to perform such a strategy $\mathcal{O}$ on $H_1^*$.

We form $H$ by connecting $H_1$ and $H_2$ with a path of length $T + 3$ whose endpoints are $r$ and $s$. The path connecting $r$ and $s$ contains $T + 2$ internal vertices – we label these vertices along the path, starting at the neighbour of $r$, as $x_1, \ldots, x_{T+2}$.

We clean $H^*$ with $V_{H^*} \setminus V_H = \{u\}$ by proceeding in the following manner:

1. We first perform the zero-visibility strategy on $H_2^*$. Because this strategy satisfies (i) and (ii), it successfully cleans $H_2^*$.

2. One cop vibrates on the edge $su$ while the other moves first to $x_{T+1}$ and then moves along the path from $x_T$ to $x_1$ (this cop does not move onto $r$ at this point).

3. Both cops move onto $u$; after this move, $x_1$ is recontaminated.

4. The cops move from $u$ to their starting positions for the strategy on $H_1^*$; after this move, $x_1$ and $x_2$ are recontaminated.

5. The cops then perform the strategy $\mathcal{O}$ on $H_1^*$; after these $T$ moves, the vertices $x_1, \ldots, x_{T+2}$ are recontaminated while the vertex $s$ and all vertices of $H_2$ are still cleaned. During the course of cleaning $H_1^*$, the vertex $u$ is visited at least every second turn.

6. At this point there is either a cop $i$, $i \in \{1, 2\}$, that has been vibrating on the edge $ru$ since $r$ was first cleaned or a cop $i$, $i \in \{1, 2\}$, has just moved onto $r$ for the first time. In either case, the cop $i$ continues/begins vibrating on the edge $ru$ so that the conditions (i) and (ii) are met. The other cop moves to $u$. After this turn, $s$ is recontaminated and the other vertices of $H_2$ are cleaned.

7. The cop $i$ vibrates on $ru$; the other one moves from $u$ onto $s$, cleaning $s$. After this move only the vertices $x_1, \ldots, x_{T+2}$ are contaminated.

8. While the cop $i$ continues vibrating on the edge $ru$, the second moves along the path connecting $s$ and $r$.

This successfully cleans $H^*$ in such a way that conditions (i) and (ii) are met.

We note that if $G$ is a tree which contains two or more vertices and $H$ is a subdivision of $G$, then $H^*$ must contain a cycle of length at least four. So, $H^*$ cannot be cleaned by a single cop. This construction produces graphs with zero-visibility copnumber exactly equal to 2. □□
Remark 4  The inequality $pw(G) \leq 2mc_0(G) - 1$ in Theorem 7 is the best possible.

Due to Theorem 2 (see [26, 27]),

$$pw(K_{2m}) = 2m - 1 = 2mc_0(K_{2m}) - 1.$$  

Thus, the bound is tight on complete graphs of even order. Moreover, Theorem 8 shows that we cannot replace $mc_0(G)$ with $c_0(G)$ in this inequality. In fact, as long as the pathwidth of a graph is 2 or greater, no upper bound for the pathwidth can be deduced from the zero-visibility copnumber.

Theorem 8  For any positive integer $k$, there is a graph $G$ with $c_0(G) = 2$ and $pw(G) \geq k$.

Proof: This follows from Lemma 6 along with the fact that if the graph $G$ is a minor of $H$, then $pw(G) \leq pw(H)$.

We note that for any graph $H$, $H$ is a minor of $H^*$, implying that $pw(H) \leq pw(H^*)$. As well, if $H$ is a subdivision of $G$, then $G$ is a minor of $H$, in which case $pw(G) \leq pw(H)$.

Let $G$ be a tree with $pw(G) \geq k$; for example, $G = T_{2k-1}$ from Example 2. We form a subdivision $H$ of $G$ such that $c_0(H^*) = 2$. We then have

$$pw(H^*) \geq pw(H) \geq pw(G) \geq k.$$  

The graph $H^*$ satisfies the statement of the theorem. \qed

5  Conclusion

There remains a considerable amount of further work concerning the zero-visibility model to be accomplished. Characterisations of $c_0$ and $mc_0$ over well-known families of graphs (such as trees, unicyclic graphs, planar graphs, series parallel graphs, etc.) are of interest. An analysis of the algorithmic complexity of accomplishing a successful zero-visibility search would cement this model’s position in the overall area of pursuit-evasion games and width parameters.

It would be very interesting to construct some sort of relationship between the value $mc_0(G) - c_0(G)$ (or possibly $mc_0(G)/c_0(G)$) and combinatoric or connective properties of the graph. In general, $c_0$ and $mc_0$ can differ by arbitrarily large amounts, but it would be interesting to characterise those graphs $G$ for which $c_0(G) = mc_0(G)$, for example.

The fact that the monotonic zero-visibility copnumber can be bounded both above and below by positive multiples of the pathwidth suggests that, in a sense, node-search and the monotonic zero-visibility search are variations of the same game. Each number is an approximation of the other, suggesting that efficient strategies in one game can usually be translated to efficient strategies in the other.

However, Theorem 8 shows that the zero-visibility copnumber can be entirely unrelated to the other two parameters. The general zero-visibility search can be carried out using methods that will not work in a node-search. The zero-visibility search is genuinely distinct from other pursuit-evasion games and informs us of different structural properties of a graph.
References


