Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

To satisfy impatient Web surfers is hard ☆

Fedor V. Fomin [a], Frédéric Giroire [b], Alain Jean-Marie [c], Dorian Mazauric [b,*], Nicolas Nisse [b]

[a] Department of Informatics, University of Bergen, Norway
[b] COATI, INRIA, I3S (CNRS/Univ. Nice-Sophia Antipolis), France
[c] INRIA and LIRMM (CNRS/Univ. Montpellier 2), France

ARTICLE INFO

Article history:
Received 6 October 2012
Received in revised form 17 December 2013
Accepted 11 January 2014
Communicated by C. Kaklamanis

Keywords:
Prefetching
Cops and robber games
PSPACE-complete
Interval graphs

ABSTRACT

Prefetching is a basic mechanism for faster data access and efficient computing. An important issue in prefetching is the trade-off between the amount of network's resources wasted by the prefetching and the gain of time. For instance, in the Web, browsers may download documents in advance while a Web surfer is surfing. Since the Web surfer follows the hyperlinks in an unpredictable way, the choice of the Web pages to be prefetched must be computed online. The question is then to determine the minimum amount of resources used by prefetching that ensures that all documents accessed by the Web surfer have previously been loaded in the cache.

We model this problem as a two-player game similar to Cops and Robber Games in graphs. Let $k \geqslant 1$ be any integer. The first player, a *fugitive*, starts on a marked vertex of a (di)graph G. The second player, an *observer*, marks at most k vertices, then the fugitive moves along one edge/arc of G to a new vertex, then the observer marks at most k vertices, etc.

The fugitive wins if it enters an unmarked vertex, and the observer wins otherwise. The *surveillance number* of a (di)graph is the minimum k such that the observer marking at most k vertices at each step can win against any strategy of the fugitive. We also consider the connected variant of this game, i.e., when a vertex can be marked only if it is adjacent to an already marked vertex.

We study the computational complexity of the game. All our results hold for both variants, connected or unrestricted. We show that deciding whether the surveillance number of a chordal graph is at most 2 is NP-hard. We also prove that deciding if the surveillance number of a DAG is at most 4 is PSPACE-complete. Moreover, we show that the problem of computing the surveillance number is NP-hard in split graphs. On the other hand, we provide polynomial time algorithms computing surveillance numbers of trees and interval graphs. Moreover, in the case of trees, we establish a combinatorial characterization of the surveillance number.

1. Introduction

Prefetching is a basic technique in computer science. It exploits the parallelism between the execution of one task and the transfer of information necessary to the next task, in order to reduce waiting times. The classical instance of the problem

☆ This work has been done during the visit of the first author at the team-project COATI, INRIA, I3S, Sophia Antipolis, France. A short version of this paper has been presented in the 6th International Conference on FUN with Algorithms [11].

* Corresponding author.

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2014.01.009

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

occurs in CPU, where instructions and data are prefetched from the memory while previous instructions are executed. The modern instance occurs in the Web, where browsers may download documents connected to the currently viewed document (Web page, video, etc.) while it is being read or viewed. Accessing the next document appears to be instantaneous to the user, and gives the impression of a large navigation speed [8]. For this reason, link prefetching has been proposed as a draft Internet standard by Mozilla [29]. However, prefetching all documents that can be accessed in the current state may exceed networking capacities, or at least, result in a waste of bandwidth since most of the alternatives will not be used. Hence, it is necessary to balance the gain of time against the waste of networking resources. Local storage memory is also a potential issue, and prefetching is classically associated with the question of cache management. However, memory in modern computers is not scarce anymore, which makes network resources the critical ones.

The models developed so far in the literature to study prefetching problems are based on the *execution digraph*, where the nodes represent the tasks (e.g., Web pages) and arcs model the fact that a task can be executed once another has been done (e.g., arcs represent hyperlinks that can be followed from a Web page). The execution of the program or the surfing of the Web then corresponds to a path in the execution digraph. The quantitative optimization of prefetching will then be based on some cost function defined on paths, reflecting for instance the inconvenience of waiting for some information while executing the tasks or surfing the Web, and possibly taking into account the consumption of network or memory resources. The related dimensioning problem consists in determining how much network bandwidth should be available so that the prefetching performance stays within some predetermined range.

It is quite likely that such optimization problems are very difficult to solve exactly. For instance, in *Markovian* models [21], where arcs of the execution digraph are associated with transition probabilities (modeling a random Web surfer), the prefetching problem can then be cast as an optimization problem in the Stochastic Dynamic Programming framework [20,24]. Its exact solution requires a computational effort which is exponential with respect to the number of nodes in the execution digraph: this is the size of the state space of these Markov Decision models.

As a first step in the analysis of prefetching optimization, we therefore consider the following simpler problem. We consider a surfer evolving over the execution digraph, and we are concerned with *perfect* prefetching, i.e., ensuring that the Web surfer never accesses a document that has not been prefetched yet. In other words, the surfer is "impatient" in the sense that it does not tolerate waiting for information. Due to network's capacity (bandwidth) limitation, it is important to limit the number of Web pages that can be prefetched at each step: We aim at determining its minimum value. In addition to being simpler than a fully specified optimization problem, this question does not need specific assumptions on the behavior of the Web surfer as in [20,24].

Let D be an execution digraph and let $v_0 \in V(D)$ be a node corresponding to the Web page from which the surfer starts. At each *step*, some amount of Web pages are prefetched and then the surfer either moves along an arc to an out-neighbor of its current position, or skips its move. The *surveillance number* of D starting in v_0 is the minimum integer k such that prefetching at most k Web pages at each step guarantees the Web surfer never waits (whatever the surfer does).

1.1. Our results

We model the above prefetching problem as a two-player game similar to Cops and Robber game (e.g., see [3,9,11,25,26]). We first prove that "monotonicity does not help", that is, if the fugitive follows only induced paths, the smallest k such that there is a winning k-strategy is equal to the surveillance number. We prove that the problem of deciding whether the surveillance number of a chordal graph is at most 2 is NP-hard. In particular, this shows that the decision problem associated with the surveillance number is not Fixed Parameter Tractable. Then, we show that computing the surveillance number is NP-hard in split graphs, a subclass of chordal graphs. In the case of digraphs, we show that deciding if the surveillance number of a DAG is at most 4 is PSPACE-complete. We propose an exact exponential-time algorithm to compute the surveillance number in general (di)graphs.

On the other hand, we provide polynomial time algorithms that compute the surveillance number and a corresponding optimal strategy in trees and interval graphs. Moreover, in the case of trees, we establish a combinatorial characterization of the surveillance number. Specifically, we show that the surveillance number of a tree T starting in $v_0 \in V(T)$ equals $\max_S \lceil \frac{|N[S]|-1}{|S|} \rceil$, where S is taken among all subtrees of T containing v_0 and $N[S]$ denotes the closed neighborhood of S. We conclude with several open questions.

1.2. Cops and Robber games

Two-player turn-by-turn games in graphs have been widely studied in the literature (e.g. Maker–Breaker games, Avoider–Enforcer games, Cops and Robber games). The game we consider here is similar to Cops and Robber games because our surfer aims at escaping the observer as the robber does in Cops and Robber games. In the initial variant of these games [25,26], one cop is placed at a vertex of a graph, then the robber chooses one vertex to be placed on, and then the players move their token along the edges of the graph, alternately starting with the cop. The cop wins if at some step of the game it occupies the same vertex as the robber. In [1], the Cop Player is allowed to use a team of $k \geqslant 1$ cops. One optimization problem is then to decide the cop number of a graph G, i.e., the minimum number of cops that are required to capture the robber in G. It is known to be W[2]-hard in general [13,18]. Lower and upper bounds on the cop number of various classes of graphs have been proved [4,5,16,28].

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

Several variants have been studied such as when the cops and the robber have different speeds [2,7,13,23], when the robber can be captured at some distance [6], when each cop can be moved a bounded number of times [15], etc. In the variant proposed in [12,14], the goal for the cops is to *guard* some part of a graph, i.e., to prevent the robber to reach some particular vertices in the graph. Eternal dominating set and eternal vertex cover can also be viewed as cops and robber games, where the robber has no token but can *attack* a vertex (or an edge) at each step and the cop must move its tokens in response to the attack [10,19].

2. Preliminaries

In this section, we formally define the problems we consider and present the notations used throughout the paper. We also present some basic results.

For any (di)graph $G = (V, E)$ considered in this paper, when $v_0 \in V$ is fixed as the starting vertex, we assume that, for any $v \in V$, there is a (directed) path from v_0 to v. In particular, if G is an undirected graph, we assume that G is connected.

Let $\Delta(G)$ be the maximum degree of G (we denote it by Δ when no ambiguity occurs). If G is a digraph, we denote by $\Delta^+(G)$ its maximum out-degree.

For any undirected graph $G = (V, E)$ and any $S \subseteq V$, let $G[S]$ be the subgraph induced by S in G. The *open neighborhood* $N(S)$ of set S is the set of vertices in $V \setminus S$ having a neighbor in S and the *closed neighborhood* of S, denoted by $N[S]$, is defined as $N(S) \cup S$. If $S = \{v\}$, we use $N(v)$ and $N[v]$ instead of $N(\{v\})$ and $N[\{v\}]$. Similarly, in a directed graph $D = (V, E)$, $N^+[S]$ denotes the closed out-neighborhood of $S \subseteq V$, i.e., the set of vertices that are in S or are an out-neighbor of a vertex in S.

A graph is a tree if it has no cycle as a subgraph. A digraph is a directed acyclic graph (*DAG*) if it has no directed cycle as a subgraph. A graph is *chordal* if it does not contain an induced cycle of length at least 4. A graph $G = (V, E)$ is a *split graph* if there is a partition (A, B) of V such that A induces a clique and B induces an independent set. Finally, G is an *interval graph* if V is a set of real intervals and two vertices are adjacent if their corresponding intervals intersect.

2.1. The surveillance game

The *surveillance problem* deals with the following two-player game in an n-node (di)graph $G = (V, E)$ with a given starting vertex $v_0 \in V$. There are two-player, *fugitive* and *observer*. The fugitive wants to escape the control of an observer whose purpose is to keep the fugitive under constant surveillance.

Let $k \geqslant 1$ be a fixed integer.

The game starts when the fugitive stands at v_0 which is initially *marked*. All nodes of G but v_0 are initially not marked. Then, turn by turn, the observer controls, or *marks*, at most k vertices and then the fugitive either moves along an edge to a (out-) neighbor of its current position, or skips its move. In particular, at every step of the game, the observer enlarges the observable part of the graph by adding to it at most k vertices. His task is to ensure that the fugitive is always in the observable area. Note that, once a vertex has been marked, it remains marked until the end of the game. The fugitive wins if, at some step, it reaches an unmarked vertex; and the observer wins otherwise. That is, the game ends when either the fugitive enters an unmarked vertex (and then the fugitive wins) or all vertices have been marked (and then observer wins).

A *configuration* of the game consists of a pair (M, f) where $v_0 \in M \subseteq V$ represents the set of the vertices that have already been marked (containing v_0) and $f \in M$ corresponds to the current position of the fugitive. A *k-strategy* (for the observer) is a function σ that assigns to any configuration C the set $\sigma(C)$ of at most k vertices that must be marked by the observer in this configuration. Note that, at each step, the observer has interest to mark as many unmarked vertices as possible.

More formally, a configuration is a pair (M, f) with $v_0 \in M \subseteq V$ and $f \in M$. A k-strategy is a function $\sigma : 2^V \times V \to 2^V$ that assigns, to any configuration (M, f), a subset $S = \sigma(M, f) \subseteq V \setminus M$ such that $|S| = \min\{k, |V \setminus M|\}$.

A k-strategy is *winning* if the observer using that strategy wins whatever be the walk followed by the fugitive starting in v_0. In other words, a strategy σ is winning if $N(f) \setminus M \subseteq \sigma(M, f)$ for any configuration (M, f) that is realizable starting from $(\{v_0\}, v_0)$ with the observer following σ.

The *surveillance number* of G starting from v_0, denoted by $sn(G, v_0)$, is the smallest k such that there is a winning k-strategy in G starting from v_0.

In the surveillance game, the fugitive plays the role of the Web surfer moving in the execution (di)graph while the observer must prefetch the Web pages before the fugitive reaches them. Before going further, we discuss some hypotheses of our model.

First, we assume a constant prefetching time for all the Web pages. It is however not a strong assumption since the surveillance game may also model the fact that some Web pages are heavier than others. Indeed, let us assume that each Web page u has a proper size $\mathcal{W}(u)$ and so a proper prefetching time, assumed to be an integer. Consider the graph G^p obtained by replacing any node u of G by a clique K_u of size $\mathcal{W}(u)$ and any edge $\{u, v\}$ by a complete bipartite graph between K_u and K_v. Thus, the surveillance problem for the weighted graph G is equivalent to the problem in G^p.

Another assumption of our model is that Web-pages are all equivalent, in the sense that the Web surfer does not spend more time on some pages than on other pages. We actually assume that the step duration is the minimum visiting time among all pages. If there exists a perfect prefetching strategy with this constant duration time, then this strategy is also

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

a perfect prefetching strategy with the initial visiting times for all the pages. This hypothesis corresponds to studying the worst case in which the visiting time of all pages is constant (and so corresponds to the minimum visiting time among all the pages).

Finally, we implicitly assume that all prefetched pages fit in the memory. This assumption is discussed in the conclusion.

2.2. Connectivity and bounds

In this section, we define a variant of the game by introducing new natural constraints and prove basic results.

In the *connected* variant of the surveillance game, the observer must mark only vertices that have neighbors already marked. In other words, the set of marked vertices must always induce a connected subgraph. A connected strategy σ is a strategy with the additional constraint that $\sigma(M, f) \cup M$ must induce a connected subgraph for any connected subset $M \subseteq V$ containing v_0. Note that it is not required that $\sigma(M, f)$ induces a connected subgraph. Let $csn(G, v_0)$ be the smallest k such that there is a winning connected k-strategy in G when the fugitive starts from v_0.

We first show that imposing the connectedness of a strategy is a strong constraint.

Lemma 1. *Let $k \geqslant 2$. There exist a graph G and a vertex $v_0 \in V(G)$ such that $csn(G, v_0) > sn(G, v_0) = k$.*

Proof. Let $k \geqslant 2$. Let G be the graph with $6k$ vertices, built as follows: a path (v_0, v_1, v_2) then $2k$ vertices a_i and b_i, $1 \leqslant i \leqslant k$, such that a_i is adjacent to v_2 and b_i, and finally a set K of $4k-3$ vertices each of which is adjacent to all b_i, $i \leqslant k$. Then $k = sn(G, v_0) < csn(G, v_0) = k+1$.

Indeed, the following k-strategy is winning: at each step, the observer marks all $i \geqslant 0$ unmarked neighbors of the current position of the fugitive, and then marks $k-i$ vertices in K. Hence $sn(G, v_0) \leqslant k$.

On the other hand, in the connected variant, at least 4 vertices, say $\{v_1, v_2, a_1, b_1\}$, must be marked before at least one vertex in K is marked. The fugitive first goes to v_1 and v_2. Then if a vertex a_i is unmarked, the fugitive goes to it and wins. Otherwise, it goes to a_2 and then b_2. At the fifth turn of the fugitive, at least $k+4$ vertices not in K must have been marked (that is the set of vertices $\{v_1, v_2, a_1, \ldots, a_k, b_1, b_2\}$). Then, when at most k vertices can be marked per step, at most $5k - (k+4) = 4k-4$ vertices of K have been marked. Thus, the fugitive can win reaching an unmarked vertex in K. Hence $csn(G, v_0) > k$.

Finally, it is easy to show that $csn(G, v_0) \leqslant k+1$ and that $sn(G, v_0) > k-1$. □

Question 1. *Does there exist a constant bounding the ratio (or the difference) between csn and sn in all graphs?*

The surveillance number of a graph is constrained by the degrees of its vertices. More precisely:

Claim 2. *For any (di)graph G with maximum (out-)degree $\Delta^{(+)}$ and for any v_0 with (out-)degree $deg^{(+)}(v_0)$, we have $deg^{(+)}(v_0) \leqslant sn(G, v_0) \leqslant csn(G, v_0) \leqslant \Delta^{(+)}$. Moreover, in undirected graphs, $csn(G, v_0) = \Delta$ if, and only if, v_0 has degree Δ.*

Proof. Clearly, $sn(G, v_0) \geqslant deg^{(+)}(v_0)$ and by definition $sn(G, v_0) \leqslant csn(G, v_0)$. On the other hand, the following $\Delta^{(+)}$-strategy is winning for the observer. At each step, the observer simply marks all unmarked (out-)neighbors of the current position of the fugitive. Hence, $csn(G, v_0) \leqslant \Delta^{(+)}$. Moreover, in the undirected case, the fugitive always arrives to any vertex (but v_0) through a neighbor already marked. Hence, following the previous strategy, the observer marks at most $\Delta - 1$ vertices at each step, except the first one. So, if $deg^{(+)}(v_0) < \Delta$ then we get that $csn(G, v_0) < \Delta$. □

The next lemma is a straightforward consequence of the previous claim.

Lemma 3. *Let G be a connected undirected graph with maximum degree $\Delta \leqslant 3$ and at least one edge. Then, $1 \leqslant csn(G, v_0) = sn(G, v_0) \leqslant 3$ and*

- *$csn(G, v_0) = sn(G, v_0) = 1$ iff G is a path, where v_0 has degree one;*
- *$csn(G, v_0) = sn(G, v_0) = 3$ iff v_0 has degree 3.*

Thus, the problem of computing the surveillance number of a graph with maximum degree at most 3 is trivial.

Question 2. *What is the complexity of computing the surveillance number in the class of graphs with maximum degree 4? with bounded degree?*

The proof of the following lemma is also straightforward.

Lemma 4. *Let G be an undirected graph with a universal vertex. For any $v_0 \in V(G)$, we have $sn(G, v_0) = csn(G, v_0) = \max\{deg(v_0), \lceil \frac{n-1}{2} \rceil\}$.*

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

2.3. The monotone variant of the game

Finally, we define a restriction of the game that will be useful throughout this paper.

In the *monotone* variant of the surveillance game, the fugitive is restricted to move at every step and to follow only induced paths in G. That is, for all $\ell > 0$, after having followed a path $(v_0, \ldots, v_\ell)$, the fugitive is not allowed to reach a vertex in $N[\{v_0, \ldots, v_{\ell-1}\}]$. Note that if the fugitive cannot move, then it loses. Let $msn(G, v_0)$ be the smallest k such that there is a winning monotone k-strategy in G when the fugitive starts from v_0, i.e., the observer can win, marking at most k vertices at each step, against a fugitive constrained to follow induced paths.

The monotone game is easier to analyze. Furthermore, we now prove that "monotonicity does not help", that is, for any graph G and $v_0 \in V(G)$, $msn(G, v_0) = sn(G, v_0)$. In other words, if the fugitive follows only induced paths, no k-strategy can be a winning (monotone) strategy, for any $k < sn(G, v_0)$. This means that in the following proofs, we can always consider that the fugitive follows induced paths, and so that the fugitive has to move at every step because an induced path is necessarily a simple path.

To prove the announced result, we give an alternative definition of a winning strategy in terms of trees reminiscent of those used in the decomposition of graphs.

First, we give some intuition about the proof. Consider a monotone k-strategy σ for the observer. We turn this strategy into a non-monotone k-strategy handling all possible trajectories (not only along induced paths) of the fugitive. To do so, while the fugitive follows an induced path, the observer uses σ. Now, assume that the fugitive follows a walk $W = (v_0, v_1, \ldots, v_m)$ and then moves to v_{m+1} which is a neighbor of v_j for $0 \leqslant j \leqslant m-1$. Intuitively, we can find a subset of $V(W)$ inducing an induced path P from v_0 to v_j and then to v_{m+1}, and such that the observer can apply the strategy σ as if the fugitive had followed P.

Recall that a strategy is defined by a function $\sigma : 2^V \times V \to 2^V$, where $|\sigma(M, f)| \leqslant k$ for any $M \subseteq V$, $f \in V$ and $\sigma(M, f)$ represents the set of vertices that must be marked when the fugitive is in f and the vertices in M have already been marked. Clearly, such a strategy can be viewed as a *decision-tree*, where each vertex of this decision tree represents a path that has been followed by the fugitive.

We first describe a tree-structure to represent the paths of G, starting from v_0. An *internal vertex* of a rooted tree is a vertex with at least one child, other vertices are called the *leaves*.

Definition 1. Let G be a (di)graph and $v_0 \in V(G)$. A *path-tree* is a pair (T, ω), where T is a tree rooted at $r \in V(T)$ and $\omega : V(T) \to V(G)$ is a mapping from the vertices of the tree to that of the (di)graph, such that $\omega(r) = v_0$ and any internal vertex $t \in V(T)$ has $\ell = |N^{(+)}[\omega(t)]|$ children $\{t_1, \ldots, t_\ell\}$ with $\{\omega(t_1), \ldots, \omega(t_\ell)\} = N^{(+)}[\omega(t)]$.

In a path-tree T, any vertex $t_i \in V(T)$ $(i \geqslant 0)$, where $(r = t_0, t_1, \ldots, t_i)$ is the path from r to t_i in T represents the walk $P_{t_i} = (v_0 = \omega(r), \omega(t_1), \ldots, \omega(t_i))$ in G. The next structure restricts the paths we want to represent to the induced paths of G starting from v_0.

Definition 2. Let G be a (di)graph and $v_0 \in V(G)$. An *induced path-tree* is a pair (T, ω), where T is a tree rooted at r and $\omega : V(T) \to V(G)$ such that $\omega(r) = v_0$ and, for any internal vertex $t_i \in V(T)$, where $(r = t_0, t_1, \ldots, t_i)$ is the path from r to t_i in T and $N = N_G^{(+)}(\omega(t_i)) \setminus N_G^{(+)}[\{\omega(t_0), \omega(t_1), \ldots, \omega(t_{i-1})\}]$, then t_i has $\ell = |N|$ children $\{u_1, \ldots, u_\ell\}$ with $\{\omega(u_1), \ldots, \omega(u_\ell)\} = N$.

Definition 3. A (*monotone*) *k-decision-tree* (k-DT) rooted at v_0 of a (di)graph G is a triple (T, ω, M) defined as follows. (T, ω) is a (induced) path-tree rooted at r and $M : V(T) \to 2^{V(G)}$ and the following properties are satisfied: $v_0 \in M(r)$ and, for any vertex $t_i \in V(T)$, where $(r = t_0, t_1, \ldots, t_i)$ is the path from r to t_i in T,

- $|M(t_i) \setminus \{v_0\}| \leqslant k$;
- for any child t of t_i, $\omega(t) \in \bigcup_{j \leqslant i} M(t_j)$;
- t_i is a leaf iff
  - In a k-decision-tree: $\bigcup_{j \leqslant i} M(t_j) = V(G)$;
  - In a monotone k-decision-tree: either $\bigcup_{j \leqslant i} M(t_j) = V(G)$ or $\omega(t_0), \omega(t_1), \ldots, \omega(t_i)$ is a maximal induced path in G.

The (induced) path-tree allows to represent all walks (induced paths) starting in v_0 in G. Namely, given $t_i \in V(T)$ with $P = (r, t_1, \ldots, t_i)$ the path in T from r to t_i, t_i represents the (induced) path $P_{t_i} = (v_0 = \omega(r), \omega(t_1), \ldots, \omega(t_i))$ in G. Moreover, for any $t \in V(T)$, the *bag* $M(t)$ represents the subset of vertices that must be marked at the step after the fugitive has followed the path P_t in G. As a consequence of the properties stated in Definition 3, no more than k vertices are marked at each step and no path in G may allow the fugitive to avoid marked vertices.

Decision-trees should be more constrained to express that it is useless to mark several times the same vertex or not to mark the maximum number of vertices at each step.

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

Definition 4. A (monotone) k-decision-tree (T, ω, M) is said to be *refined* if

- for any internal vertex $t \in V(T)$, $|M(t) \setminus \{v_0\}| = k$;
- for any vertex $t_i \in V(T)$, where $(r = t_0, t_1, \ldots, t_i)$ is the path from r to t_i in T, $M(t_i) \subseteq V(G) \setminus \bigcup_{j<i} M(t_j)$.

Recall that the height of a rooted tree is the maximum length (number of edges) of a path between the root and a leaf of the tree. Note that a refined k-DT of n-node graph G has height at most $\lceil \frac{n-1}{k} \rceil$.

Lemma 5. *If G admits a (monotone) k-decision-tree rooted at v_0, then G admits a (monotone) refined k-decision-tree rooted at v_0.*

Proof. We show that every decision-tree that is not refined can be transformed into one "more refined". Iterating the process ends with a (monotone) refined k-decision-tree rooted at v_0.

Let (T, ω, M) be a (monotone) k-DT rooted at v_0. Among the paths $(r = t_0, t_1, \ldots, t_i)$ in T, that do not satisfy the conditions of a refined k-DT, pick one for which t_i is the closest from r, the root of T. If $M(t_i) \cap \bigcup_{j<i} M(t_j) \neq \emptyset$, then replace $M(t_i)$ with $M(t_i) \setminus \bigcup_{j<i} M(t_j)$. Otherwise, let $v \in V(G) \setminus \bigcup_{j \leqslant i} M(t_j)$ and replace $M(t_i)$ with $M(t_i) \cup \{v\}$. Finally, if $\bigcup_{j \leqslant i} M(t_j) = V(G)$, remove from T all subtrees rooted at a child of t_i. The path now satisfies the conditions of a refined DT. □

Lemma 6. *A (di)graph G admits a (respectively, monotone) k-decision-tree rooted at v_0 if, and only if, $sn(G, v_0) \leqslant k$ (respectively, if, and only if, $msn(G, v_0) \leqslant k$).*

Proof. Assume $sn(G, v_0) \leqslant k$ (respectively, $msn(G, v_0) \leqslant k$) and let σ be a k-strategy such that $\sigma(M, f) \subseteq V(G) \setminus M$, and $|\sigma(M, f)| = k$ or $M \cup \sigma(M, f) = V(G)$ for any $M \subseteq V(G)$, $f \in M$. Let (T, ω) be the (induced) path-tree representing all walks (or induced paths) of G starting in v_0 and of length at most $\lceil \frac{|V(G)|-1}{k} \rceil$. Then, let $M(r) = \{v_0\} \cup \sigma(\{v_0\}, v_0)$, and, for any vertex $t_i \in V(T) \setminus \{r\}$, where $(r = t_0, t_1, \ldots, t_i)$ is the path from r to t_i in T, let us define $M(t_i) = \sigma(\bigcup_{j<i} M(t_j), \omega(t_i))$. All conditions of Definition 3 are satisfied and thus (T, ω, M) is a (monotone) k-DT rooted at v_0. In particular, the third condition is satisfied because of the height of T and the fact that σ marks as many unmarked nodes as possible at each step.

Let (T, ω, M) be a k-DT rooted at v_0. Let $\sigma : 2^{V(G)} \times V(G) \to 2^{V(G)}$ be any application satisfying that, for any $t_i \in V(T)$, where $(r = t_0, t_1, \ldots, t_i)$ is the path from r to t_i in T, $\sigma(\bigcup_{j<i} M(t_j), \omega(t_i)) = M(t_i) \setminus \{v_0\}$. Then, σ is a winning k-strategy. □

Now, we can prove the main result of this section.

Theorem 7. *For any (di)graph G and $v_0 \in V(G)$, $sn(G, v_0) = msn(G, v_0)$.*

Proof. By definition, $sn(G, v_0) \geqslant msn(G, v_0)$.

If $msn(G, v_0) \leqslant k$, by Lemma 6, there is a monotone k-DT of G rooted at v_0. By Lemma 5, there is a refined monotone k-DT of G rooted at v_0. We show that a k-DT rooted at v_0 of G can be built from any refined monotone k-DT (T, ω, M) of G rooted at v_0. Then, by Lemma 6, $sn(G, v_0) \leqslant k$.

Let (T, ω, M) be a refined monotone k-DT of G rooted at v_0. If (T, ω, M) is a k-DT, we are done. Assume therefore that (T, ω, M) is not a k-DT. This means that (T, ω) is an induced path-tree and not a path-tree. In other words, there is a vertex $t \in V(T)$ with children $(u_1, \ldots, u_\ell)$ such that there is, in G, a neighboring vertex $y \in N_G[\omega(t)]$ and, for all $i \leqslant \ell$, $\omega(u_i) \neq y$. We say that such a vertex t satisfies property $\mathcal{P}$, with y a "bad neighbor". Consider one such vertex closest to r. Two cases are to be considered according to whether the bad vertex y is $\omega(t)$ or in $N_G(\omega(t))$.

- Assume first $y = \omega(t)$. Let S be the subtree of T rooted at t. We transform (T, ω, M) into (T', ω', M') by adding the sub-decision-tree "induced" by S as a child of t in T. That is: T' is obtained from T by adding a copy of S with its root adjacent to t in T. Then, for any $z \in V(T') = V(T) \cup V(S)$, $\omega'(z) = \omega(z)$ and $M'(z) = M(z)$. Then, let (T^*, ω^*, M^*) be obtained by refining the resulting path-tree, i.e., by applying to (T', ω', M') the process described in the proof of Lemma 5.
- Now, assume that $y \in N_G(\omega(t))$. Let $(r = t_0, t_1, \ldots, t_i = t)$ be the path from r to t in T. Since $y \notin \{\omega(u_j) : j \leqslant \ell\}$, by definition of monotone decision-trees, it means that there is $j < i$ such that t_j has a child s and $\omega(s) = y$. Let S be the subtree of T rooted at s. We transform (T, ω, M) by adding the sub-decision-tree "induced" by S as a child of t in T. Then, we refine the obtained decision-tree.

The process consists in repeating the transformation while there is a node t that satisfies property $\mathcal{P}$. Note that, while there still are some vertices satisfying $\mathcal{P}$, the resulting structure is neither a decision-tree nor a monotone decision-tree since the tree T^* of this structure is neither a path-tree nor an induced path-tree (actually, it is "between" a path-tree and an induced path-tree). However, all other properties of a decision-tree remain satisfied.

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

We finally show that a finite number of such transformations is sufficient to obtain a decision-tree. Indeed, it is sufficient to remark that the following "potential function" Φ strictly decreases each time the transformation is applied. Moreover, the size of the structure remains bounded in the size of the initial monotone decision-tree (T, ω, M) since we consider only refined versions.

Let $\Phi(T, \omega, M)$ be the sum of the values $\phi(d(t,r)) \times (|N_G^{(+)}[\omega(t)]| - |\{\omega(s)\colon s \text{ child of } t \text{ in } T\}|)$, over all vertices t in $V(T)$, where $d(t,r)$ is the distance between t and r and ϕ is any function such that for all i, $\phi(i) - \Delta^{\lceil \frac{n-1}{k} \rceil - i + 1}\phi(i+1) \geqslant c > 0$, where c is some positive constant.

Each time we apply the procedure on a node $t \in V(T)$ at distance i of the root, we add one child to t. Hence, $(|N_G^{(+)}[\omega(t)]| - |\{\omega(s)\colon s \text{ child of } t \text{ in } T\}|)$ is decreased by one and this contributes to decreasing $\Phi(T, \omega, M)$ by $\phi(i)$. However, as "child" of t, we add a subtree S with height at most $\lceil \frac{n-1}{k} \rceil - i$ (because it is refined). Therefore, S may have at most $\Delta^{\lceil \frac{n-1}{k} \rceil - i}$ vertices and the contribution of each vertex is at most $\Delta \cdot \phi(i+1)$. This contributes to increasing the global sum $\Phi(T, \omega, M)$ of at most $\Delta^{\lceil \frac{n-1}{k} \rceil - i + 1}\phi(i+1)$. Since $\phi(i) - \Delta^{\lceil \frac{n-1}{k} \rceil - i + 1}\phi(i+1) > c$, the global sum $\Phi(T, \omega, M)$ decreases by at least c at each step. Furthermore, a finite number of transformations is sufficient because for all i, $\phi(i) - \Delta^{\lceil \frac{n-1}{k} \rceil - i + 1}\phi(i+1) \geqslant c$. □

The same result holds for the connected variant. Indeed, a (monotone) decision-tree (T, ω, M) corresponds to a connected strategy if and only if for any path $(r = t_0, \ldots, t_i)$ in T, the set $\bigcup_{0 \leqslant j \leqslant i} M(t_j)$ induces a connected subgraph of G. Starting from a decision-tree that corresponds to a connected strategy, the transformation of decision-trees described in the proof of the previous theorem preserves this property (i.e., the connectivity of the strategy).

Let $mcsn(G, v_0)$ be the smallest k such that there is a winning monotone connected k-strategy in G when the fugitive starts from v_0, i.e., the observer can win, marking at most k vertices at each step such that the set of marked vertices must always induce a connected subgraph, against a fugitive constrained to follow induced paths.

Theorem 8. *For any (di)graph G and $v_0 \in V(G)$, $csn(G, v_0) = mcsn(G, v_0)$.*

3. Difficult problems

In this section, we study the computational complexity of the decision version of the problem: given a graph G with $v_0 \in V(G)$ and an integer k, the task is to decide whether $sn(G, v_0) \leqslant k$. We also consider the variant of the problem where the fugitive must win in a fixed number of steps. Moreover, for all the G graphs we consider in our reductions, we have $sn(G, v_0) = csn(G, v_0)$. Hence, our hardness results also apply to the connected variant of the problem.

We use in Theorem 9 and in Theorem 10, a reduction from the 3-Hitting Set Problem. In the 3-Hitting Set Problem, we are given a set $\mathcal{I}$ of elements, a set $\mathcal{S}$ of subsets of size 3 of $\mathcal{I}$ and $k \in \mathbb{N}$ as the input. The question is to decide whether there exists a set $H \subseteq \mathcal{I}$ of size at most k such that $H \cap S \neq \emptyset$ for all $S \in \mathcal{S}$. The 3-Hitting Set Problem is a classical NP-complete problem [17].

We start by proving that deciding whether $sn(G, v_0) \leqslant 2$ for a graph G and $v_0 \in V(G)$ is NP-hard on chordal graphs. Let us remind that a graph is *chordal* if it contains no induced cycle of length at least 4.

Theorem 9. *Deciding if $sn(G, v_0) \leqslant 2$ (respectively, $csn(G, v_0) \leqslant 2$) is NP-hard in chordal graphs.*

Proof. Let $(\mathcal{I} = \{e_1, \ldots, e_n\}, \mathcal{S} = \{S_1, \ldots, S_m\})$ and $k \geqslant 1$ be an instance of the 3-Hitting Set Problem. We construct the chordal graph G as follows. Let $P = \{v_0, \ldots, v_{m+k-2}\}$ be a path, K_m be the complete graph with vertices $\{S_1, \ldots, S_m\}$ and $e_1, \ldots, e_n$ be n isolated vertices. We add an edge from v_{m+k-2} to all vertices of K_m, and for each $i \leqslant n$ and $j \leqslant m$, add an edge between e_i and S_j if and only if $e_i \in S_j$. Clearly, G is chordal.

First, we show that, if there exists a set $H \subseteq \mathcal{I}$ of size k such that $H \cap S \neq \emptyset$ for all $S \in \mathcal{S}$, then $sn(G, v_0) \leqslant 2$. The 2-strategy of the observer first consists in marking the vertices v_1 to v_{m+k-2} in order, then the vertices of K_m and finally the vertices of H. This can be done in $m + k - 1$ steps where, at each step, all neighbors of the current position of the fugitive are marked. Because H is a hitting set of $\mathcal{S}$, after the $(m + k - 1)$-th step, each vertex S_i, $i \leqslant m$, has at most two unmarked neighbors, all other vertices have all their neighbors marked and only some vertices in $e_1, \ldots, e_n$ can be unmarked. Finally, from this step, the strategy of the observer consists in marking the unmarked neighbors of the current position of the fugitive. Clearly, the fugitive cannot win and the strategy we described is a winning 2-strategy. Note that this strategy is a connected 2-strategy. Hence, $sn(G, v_0) \leqslant csn(G, v_0) \leqslant 2$.

Now, assume that, for any $H \subseteq \mathcal{I}$ of size at most k, there is $S \in \mathcal{S}$ such that $S \cap H = \emptyset$. We show that $sn(G, v_0) > 2$ in this case. Assume that the observer can mark up to 2 vertices in each step, and we describe a winning strategy for the fugitive, which implies that $2 < sn(G, v_0) \leqslant csn(G, v_0)$. The escape strategy for the fugitive first consists of going to v_{m+k-2} (this takes $m + k - 2$ steps). Then, we may assume that after the $(m + k - 1)$-th step of the observer, all vertices of P and K_m are marked—otherwise the fugitive either would have won earlier, or can win by going to an unmarked vertex in K_m in its next move. This means that at most k of $e_1, \ldots, e_n$ has been marked up to this step. Let H denote the set of these vertices. Hence, when it is the turn of the fugitive who is occupying vertex v_{m+k-2}, there is $S_i \in V(K_m)$ with $H \cap S_i = \emptyset$,

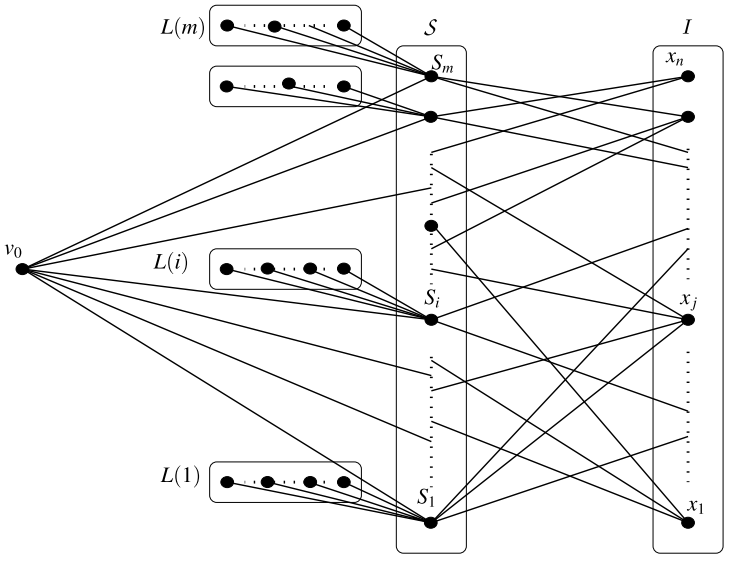

Fig. 1. Example of graph G in the reduction of the proof of Theorem 10. $L(j)$, $j \leqslant m$, represents the set of $m+k-2$ leaves adjacent to S_j.

i.e., all three neighbors of S_i are unmarked. Then, the fugitive goes to S_i. The observer marks at most 2 of the neighbors of S_i, and the fugitive can reach an unmarked vertex. Hence, $sn(G, v_0) > 2$. □

We prove that the problem of deciding whether $sn(G, v_0) \leqslant k$ is NP-hard on split graphs. Let us remind that a graph is a *split graph* if the vertices can be partitioned into a clique and an independent set. Furthermore, we prove that this NP-hardness result also holds if the number of steps is constrained to be at most 2.

Let $\ell \geqslant 1$. We define a restriction of the game where the fugitive wins if it reaches an unmarked vertex in at most ℓ steps. Let $sn(G, v_0, \ell)$ (respectively, $csn(G, v_0, \ell)$) be the smallest k such that there is a (connected) winning k-strategy in G against a fugitive starting from v_0 in this setting.

Theorem 10. *The problems of deciding whether $sn(G, v_0) \leqslant k$, or $csn(G, v_0) \leqslant k$, or $sn(G, v_0, 2) \leqslant k$, or $csn(G, v_0, 2) \leqslant k$ are NP-hard in split graphs with k as part of the input.*

Proof. Again, we reduce the 3-Hitting Set Problem to this problem. Let $(\mathcal{I} = \{x_1, \ldots, x_n\}, \mathcal{S} = \{S_1, \ldots, S_m\})$ and $k \geqslant 1$ be an instance of the 3-Hitting Set Problem.

Let us build the split graph G described in Fig. 1. Let K_{m+1} be the complete graph with vertices $\{v_0, S_1, \ldots, S_m\}$ and let $\{x_1, \ldots, x_n\}$ be n isolated vertices. For all $1 \leqslant i \leqslant m$, add $m+k-2$ extra nodes, each of degree 1, adjacent to S_i and for all $1 \leqslant j \leqslant n$ add an edge between x_j and S_i if $x_j \in S_i$. Note that, in this graph, there are no induced paths starting from v_0 and with more than two edges. Hence, by Theorems 7 and 8, $sn(G, v_0) = sn(G, v_0, 2)$ and $csn(G, v_0) = csn(G, v_0, 2)$.

As in the proof of Theorem 9, we prove that $sn(G, v_0) \leqslant k + m$ (and $csn(G, v_0) \leqslant k + m$) if and only if $(\mathcal{I} = \{x_1, \ldots, x_n\}, \mathcal{S} = \{S_1, \ldots, S_m\})$ admits a hitting set of size at most k.

Indeed, if there is a hitting set H of size k, then the observer, allowed to mark $k+m$ vertices per step, first marks all vertices of K_{m+1} (except v_0 that is already marked) and all vertices of H. At the second step, the observer marks the unmarked neighbors of the current position of the fugitive.

Conversely, assume that $\mathcal{S}$ admits no hitting set of size at most k and let us assume that the observer can mark at most $m+k$ vertices. The first move of the fugitive is to go toward a vertex S_j $(1 \leqslant j \leqslant m)$ with at least $m+k+1$ unmarked neighbors. Then, the fugitive wins after its second move. This concludes the proof. □

Next, we show that the problem of deciding whether $sn(G, v_0) \leqslant 4$ is PSPACE-complete in DAGs. In the following, we reduce the 3-QSAT problem to our problem. For a set of boolean variables $x_0, y_0, x_1, y_1, \ldots, x_n, y_n$ and a boolean formula $F = C_1 \wedge \cdots \wedge C_m$, ($C_j$ is a 3-clause), the 3-QSAT problem aims at deciding whether the expression $\Phi = \forall x_0 \exists y_0 \forall x_1 \exists y_1 \cdots \forall x_n \exists y_n F$ is true. 3-QSAT is PSPACE-complete [17].

Lemma 11. *The problem of deciding whether $sn(G, v_0) \leqslant 4$ (respectively, $csn(G, v_0) \leqslant 4$) is PSPACE-hard in DAGs.*

Proof. Let $F = C_1 \wedge \cdots \wedge C_m$ be a boolean formula with $x_0, y_0, x_1, y_1, \ldots, x_n, y_n$ as variables and

$$\Phi = \forall x_0 \exists y_0 \forall x_1 \exists y_1 \cdots \forall x_n \exists y_n F$$

be an instance of the 3-QSAT Problem. Let D be the DAG built as follows.

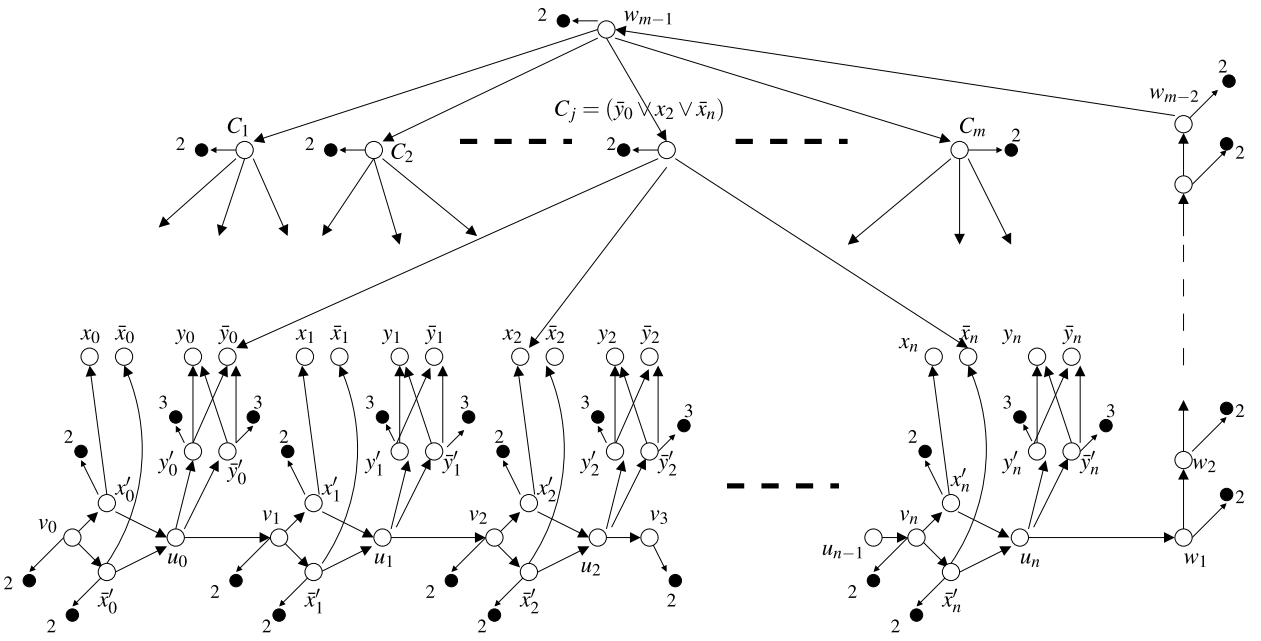

Fig. 2. Example of the reduction in the proof of Lemma 11. A black node labeled with integer i and that is the out-neighbor of a vertex v, corresponds to i leaves that are in $N^+(v)$.

We start with the set of vertices $\{u_i, v_i, x'_i, \bar{x}'_i, x_i, \bar{x}_i, y'_i, \bar{y}'_i, y_i, \bar{y}_i\}_{0 \leqslant i \leqslant n}$. For all $0 \leqslant i \leqslant n$, there are arcs from v_i to x'_i and $\bar{x}'_i$, one arc from x'_i to x_i and one arc from $\bar{x}'_i$ to $\bar{x}_i$. For all $0 \leqslant i \leqslant n$, there are arcs from x'_i and $\bar{x}'_i$ to u_i, arcs from u_i to both y'_i and $\bar{y}'_i$ and arcs from both of y'_i and $\bar{y}'_i$ to both of y_i and $\bar{y}_i$. Then, for all $0 \leqslant i < n$, there is one arc from u_i to v_{i+1}. Add the directed path $(w_1, \ldots, w_{m-1})$ with one arc from u_n to w_1 and such that w_{m-1} has m out-neighbors $C_1, \ldots, C_m$. For all $j \leqslant m$ and $0 \leqslant i \leqslant n$, add one arc from C_j to x_i (respectively, $\bar{x}_i$, y_i, $\bar{y}_i$) if x_i (respectively, $\bar{x}_i$, y_i, $\bar{y}_i$) appears in the clause C_j. Finally, for all $0 \leqslant i \leqslant n$, $k \leqslant m-1$, $j \leqslant m$ add two out-neighbors leaves to each vertex in $\{v_i, x'_i, \bar{x}'_i, w_k, C_j\}$, and, for all $0 \leqslant i \leqslant n$, add three out-neighbors leaves to each of y'_i and $\bar{y}'_i$. An example of such DAG D is depicted in Fig. 2.

Since v_0 has out-degree 4, $csn(D, v_0) \geqslant sn(D, v_0) \geqslant 4$ and the first step of the observer, endowed with 4 marks, is to mark all four out-neighbors of v_0 (the two leaves and x'_0 and $\bar{x}'_0$). We now show that $sn(D, v_0) = 4$ (and $csn(D, v_0) = 4$) if and only if Φ is true.

Intuitively, during the game, the fugitive chooses the x_i's arbitrarily, and the observer chooses the y_i's accordingly. The fugitive chooses each x_i by entering it, and the observer chooses each y_i by marking it. More precisely, the fugitive will have to follow a path from v_0 to w_1 because otherwise, we prove that it loses. During this walk, the fugitive chooses the x_i's arbitrarily: at each step it occupies some node v_i $(0 \leqslant i \leqslant n)$, it may go to x'_i which will force the observer to mark $x_i \in N(x'_i)$, or the fugitive may go to $\bar{x}'_i$ and then $\bar{x}_i$ must be marked. Moreover, our construction ensures that the observer can mark only one of x_i and $\bar{x}_i$. On the other hand, our construction ensures that, when the fugitive arrives at some node u_i $(0 \leqslant i \leqslant n)$, the observer can freely choose to mark either y_i or $\bar{y}_i$. In particular, this choice of the observer depends on the previous choices of the fugitive and of the observer. We show that, if the observer can ensure that its choices satisfy Φ then it wins, and the fugitive wins otherwise.

We first prove that the fugitive has to follow an induced path from v_0 to w_1 and that, when it arrives there, some property (see Claim 12) is satisfied. This property will allow us to prove the lemma.

Claim 12. *After the $3(n+1)$-th move of the fugitive,*

1. *the fugitive occupies vertex w_1 after having followed the directed path $P = (v_0, a'_0, u_0, v_1, a'_1, u_1, \ldots, v_n, a'_n, u_n, w_1)$, where $a'_j \in \{x'_j, \bar{x}'_j\}$ for all $0 \leqslant j \leqslant n$, and*
2. *at this step, the set of vertices marked consists of all the vertices of P and the out-neighbors of the vertices in P but $N^+(w_1)$, plus, for all $j \leqslant n$, exactly one of y_j and $\bar{y}_j$, and*
3. *for all $j \leqslant n$, the choice of which vertex has been marked in $\{y_j, \bar{y}_j\}$ depends only on the observer (not on the path followed by the fugitive).*

Proof. First, we will prove by induction on i that, after the $(3i+1)$-th step of the observer, some property $\mathcal{P}_i$ is satisfied, for all $0 \leqslant i < n$. Then, assuming that $\mathcal{P}_{n-1}$ is satisfied after the $(3(n-1)+1)$-th step of the observer, we can prove the claim using the same arguments as in the induction.

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

For all $0 \leqslant i < n$, we say that Property $\mathcal{P}_i$ is satisfied if, after the $(3i+1)$-th step of the observer:

1. the fugitive occupies vertex v_i after having followed the directed path $P = (v_0, a'_0, u_0, v_1, a'_1, u_1, \ldots, v_{i-1}, a'_{i-1}, u_{i-1}, v_i)$, where $a'_j \in \{x'_j, \bar{x}'_j\}$ for all $0 \leqslant j < i$, and
2. at this step, the set of vertices marked consists of all the vertices in P and the out-neighbors of the vertices in P plus, for all $j < i$, exactly one of y_j or $\bar{y}_j$, and
3. for all $j < i$, the choice of which vertex has been marked in $\{y_j, \bar{y}_j\}$ depends only on the observer (not on the path followed by the fugitive).

All these assumptions are satisfied for $i = 0$: $\mathcal{P}_0$ holds true. Now, let us assume that $\mathcal{P}_i$ holds for some $0 \leqslant i \leqslant n$. We show that: $\mathcal{P}_{i+1}$ holds if $i < n$, and that the claim holds if $i = n$.

Consider the game just after the $(3i+1)$-th step of the observer. Note that, by the induction hypothesis ($\mathcal{P}_i$ holds), the four out-neighbors of v_i (two of which are leaves) are marked. Since it is useless for the fugitive to remain at v_i (Theorem 7), then it goes to $a'_i \in \{x'_i, \bar{x}'_i\}$. Node a'_i has four out-neighbors that are, by the induction hypothesis, unmarked. Hence, the observer must mark these four vertices. In particular, if $a'_i = x'_i$ (respectively, if $a'_i = \bar{x}'_i$) then the observer marks $a_i = x_i$ (respectively, $a_i = \bar{x}_i$) while $\bar{x}_i$ (respectively, x_i) remains unmarked. Then, the fugitive must go to u_i since the other three out-neighbors of a'_i have no out-neighbors.

Since u_i has three out-neighbors (y'_i, $\bar{y}'_i$ and v_{i+1}), the observer must mark these three vertices. Moreover, assume that the fourth vertex marked by the observer at this step is neither y_i nor $\bar{y}_i$: then the fugitive goes to the vertex in $\{y'_i, \bar{y}'_i\}$ with still its five out-neighbors unmarked, and then the fugitive will win at the next step. Hence, the observer must mark b_i that is either y_i or $\bar{y}_i$. It is important to note that the choice of which of these two vertices is marked is completely free for the observer. After this step of the observer, both y'_i and $\bar{y}'_i$ have four unmarked out-neighbors and all the five out-neighbors of y'_i and $\bar{y}'_i$ have no out-neighbors themselves. Hence, the fugitive would lose if it went to y'_i or $\bar{y}'_i$. Hence, the fugitive must go to v_{i+1} (where v_{n+1} is set to be w_1).

If $i < n$, then v_{i+1} has exactly four out-neighbors that must be marked by the observer, and then the induction hypothesis is satisfied for $i+1$. If $i = n$, then $v_{i+1} = w_1$ and the claim holds. □

Let X be the set of vertices consisting of $\{w_2, \ldots, w_{m-1}, C_1, \ldots, C_m\}$ plus the $2(m-1)$ leaves adjacent to $w_1, \ldots, w_{m-1}$. Let Y be the set of vertices consisting of $\{x_0, \bar{x}_0, y_0, \bar{y}_0, \ldots, x_n, \bar{x}_n, y_n, \bar{y}_n\}$ plus the $2m$ leaves adjacent to the C_j's.

By Claim 12, after the $3(n+1)$-th move of the fugitive, no vertices in X are marked. Moreover, the set of marked vertices in Y is $\{a_0, b_0, \ldots, a_n, b_n\}$, where, for all $i \leqslant n$, $a_i \in \{x_i, \bar{x}_i\}$ has been imposed by the fugitive and $b_i \in \{y_i, \bar{y}_i\}$ has been chosen by the observer. In particular, if Φ is true, the observer can choose the b_i's such that $F(a_0, b_0, \ldots, a_n, b_n)$ is true whatever be the choices of the fugitive. On the other hand, if Φ is false, the fugitive can choose the a_i's such that $F(a_0, b_0, \ldots, a_n, b_n)$ is false whatever be the choices of the observer.

Now, from the $(3n+1)$-th step of the observer to the end of its $(3n+m)$-th step, the observer can mark at most $4(m-1) = |X|$ vertices. Moreover, between these steps, the fugitive must follow the path $Q = (w_1, \ldots, w_{m-1})$ (all other vertices the fugitive can access having no out-neighbors). Hence, the only choice for the observer is to successively mark all vertices in X otherwise, at some step along the path Q or just after the $(3n+m)$-th step of the observer, the fugitive could have reached an unmarked vertex. Note that the marking process can be done in a connected way.

Finally, after the $(3n+m)$-th step of the observer, the fugitive stands on w_{m-1}, all vertices in $\{C_1, \ldots, C_m\}$ are marked while the set of marked vertices in Y is $\{a_0; b_0, \ldots, a_n, b_n\}$. Now, if Φ is false, by the choice of the a_i's by the fugitive, there is a clause C_j with its five out-neighbors unmarked: the fugitive goes to C_j and will win at the next step. On the other hand, if Φ is true, by the choice of the b_i's by the observer, all C_j's have at most four unmarked out-neighbors. Whatever be the next moves of the fugitive, it will reach a marked vertex without out-neighbors.

This concludes the proof of Lemma 11. □

Lemma 13. *For every $k \geqslant 1$, the problem of deciding whether $sn(G, v_0) \leqslant k$ (respectively, $csn(G, v_0) \leqslant k$) is in PSPACE.*

Proof. The proof is similar as that of Lemma 4 in [15], so we do not go into the details. Let n be the number of vertices in graph G. Every game lasts at most n rounds. At each round, the configuration (M, f) can be encoded within polynomial space. This means that the problem is in NPSPACE (nondeterministic polynomial space)—a nondeterministic Turing machine deciding the problem uses polynomial space on every branch of its computation. By Savitch's theorem [27], the problem is in PSPACE. □

Theorem 14. *The problem of deciding whether $sn(G, v_0) \leqslant 4$ is PSPACE-complete in DAGs.*

An interesting question is to determine if the problem remains PSPACE-hard in undirected graphs. In comparison, Mamino recently proved that the cops and robber game is PSPACE-hard in undirected graphs [22].

The following theorem provides an exponential algorithm for computing $sn(G, v_0)$ (respectively, $csn(G, v_0)$). Here, we use a modified big-Oh notation that suppresses all polynomially bounded factors. For functions f and g we write $f(n) = O^*(g(n))$ if $f(n) = O(g(n)poly(n))$, where $poly(n)$ is a polynomial.

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

Theorem 15. *Given an n-node graph and a node $v_0 \in V(G)$, $sn(G, v_0)$ (respectively, $csn(G, v_0)$) can be computed in time $O^*(4^n)$.*

Proof. For each $k \geqslant 1$, we decide if $sn(G, v_0) \leqslant k$. We consider the *arena digraph* $\mathcal{G}$ whose vertices are configurations of the game, i.e., the pairs (M, f), where $v_0, f \in M \subseteq V(G)$, $N[f] \subseteq M$ and $|M \setminus \{v_0\}| = ki$ for some $i > 0$ (or $M = V(G)$). Moreover, there is an arc from (M, f) to (M', f') if $f' \in N(f)$ and $M \subset M'$ and $|M'| = |M| + k$ (or $|V(G) \setminus M| \leqslant k$ and $M' = V(G)$). Note that $\mathcal{G}$ is a DAG (because of $M \subset M'$) and that $|V(\mathcal{G})| \leqslant 2^n n$ (since there are at most 2^n choices for M and n choices for f). Hence, the amount of arcs in $\mathcal{G}$ is $O^*(4^n)$.

We consider the following labeling process. Initially, all configurations $(V(G), v)$, for all $v \in V(G)$, are labeled with $\lceil \frac{n-1}{k} \rceil$, and all other configurations are labeled with ∞. Iteratively, a configuration (M, f) with $|M| = ki + 1$ is labeled i if, for all $f' \in N_G(f)$, then $f' \in M$ and there is an out-neighbor (M', f') of (M, f) and that is labeled at most $i + 1$. We show that $sn(G, v_0) \leqslant k$ if and only if there is a configuration (M, v_0), $|M| = k + 1$, labeled with 1.

We first show by induction on i, that the observer can win starting from any configuration labeled with $\lceil \frac{n-1}{k} \rceil - i$. If $i = 0$, the result holds trivially. Assume that the result holds for some i, $0 < i < \lceil \frac{n-1}{k} \rceil - 1$. Let (M, f) be a configuration labeled with $\lceil \frac{n-1}{k} \rceil - (i + 1)$. For any $f' \in N(f)$, by definition of the labeling process, there is a configuration (M', f') out-neighbor of (M, f) and labeled with $\lceil \frac{n-1}{k} \rceil - i$. If the fugitive goes from f to f', then the observer marks the vertices in $M' \setminus M$ and the game reaches the configuration (M', f'). Hence, by the induction hypothesis, the observer wins. So, applying the result for $i = \lceil \frac{n-1}{k} \rceil - 1$, the observer wins starting from any configuration (M, v_0), $|M| = k + 1$, labeled 1. To reach this configuration, the first step of the observer is to mark the k vertices in $M \setminus \{v_0\}$. Therefore, $sn(G, v_0) \leqslant k$.

Now assume that $sn(G, v_0) \leqslant k$. Let σ be a winning k-strategy for the observer. For any walk $W = (v_0, v_1, \ldots, v_i)$ followed by the fugitive, let $M(W)$ be the set of vertices marked by the observer (using σ) after the fugitive has followed W until v_i and when it is the turn of the fugitive. By reverse induction on i, the labeling process labels $(M(W), v_i)$ with $i + 1$. This shows that $(\{v_0\} \cup \sigma(\{v_0\}, v_0), v_0)$ is labeled with 1.

For each k, the algorithm runs in time proportional to the size of $\mathcal{G}$ (number of arcs). Thus the total running time of the algorithm is $O^*(4^n)$ since at most n values of k must be tested.

To obtain the same result with $csn(G, v_0)$, it is sufficient to modify the definition of a configuration (M, f) by imposing that M must induce a connected subgraph. □

4. Polynomial-time algorithms in some graph classes

In this section, we describe polynomial-time algorithms to compute the surveillance number of trees and interval graphs. Moreover, in both these classes of graphs, we show that connectedness does not cost, in the sense that the surveillance number equals the connected surveillance number.

4.1. Keeping a tree under surveillance

We first present a polynomial-time algorithm to compute $sn(T, v_0) = csn(T, v_0)$ for any tree $T = (V, E)$ and any $v_0 \in V$. For convenience but without loss of generality, we will say that T is rooted at v_0. Recall that the height of T is the maximum length (number of edges) of a path between the root v_0 and a leaf of T. Let $k_0 \geqslant 0$.

In the following, we will use a generalization of the original game, in which v_0 plus at most k_0 other vertices chosen by the observer are initially marked. Said differently, during its first move in the game, the observer may mark $k + k_0$ vertices. Let $k \geqslant 0$. We define the function $f_k : V(T) \to \mathbb{N}$ in the following recursive way:

- $f_k(v) = 0$ for any leaf v of T;
- for any $v \in V(T)$ with d children, $f_k(v) = \max\{0, d + \sum_{w \in C} f_k(w) - k\}$, where C is the set of children of v.

Lemma 16. *Let T be a tree rooted at v_0. Then $f_k(v_0) = 0$ if, and only if, $sn(T, v_0) \leqslant k$, and if, and only if, $csn(T, v_0) \leqslant k$.*

Proof. We prove by induction on the height of T that the observer cannot win the game by marking at most k vertices per step, even if any set of at most $f_k(v_0) - 1$ vertices in $V(T) \setminus \{v_0\}$ are initially marked. Moreover, we prove that the observer can win in a connected way, marking at most k vertices per step, if some set of at most $f_k(v_0)$ vertices (chosen by the observer) plus v_0 are initially marked.

The result holds if T is reduced to one vertex. So we may assume that T has height at least 1.

If T has height 1 and v_0 has degree d, then $f_k(v_0) = \max\{0, d - k\}$ and the result holds. Indeed, if v_0 and $f_k(v_0)$ other vertices are initially marked, then during its first step, the observer marks all remaining vertices (their number is at most k) and wins. Such a strategy is clearly connected. On the other hand, if v_0 and at most $f_k(v_0) - 1$ vertices are marked, then after the first step of the observer (when he has marked k other vertices), at least one neighbor of v_0 is still unmarked. The fugitive can go there and wins.

Now, assume that the result holds for any tree of height at most $h \geqslant 1$. Let T be rooted at v_0 and be of height $h + 1$. We show that the result holds for T.

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

Let $(v_1, \ldots, v_r)$ be the children of v_0 and let T_i be the subtree of T rooted at v_i, $1 \leqslant i \leqslant r$. By the induction hypothesis, for all $1 \leqslant i \leqslant r$, there is a set $I_i \subseteq V(T_i) \setminus \{v_i\}$ of $f_k(v_i)$ vertices such that $I_i \cup \{v_i\}$ induces a subtree of T_i and, if the vertices of I_i and v_i are initially marked in T_i, then the observer can win in T_i starting from v_i, marking at most k vertices per step, and in a connected way. On the contrary, if strictly less than $f_k(v_i)$ vertices are initially marked in $V(T_i) \setminus \{v_i\}$, then the fugitive wins in T_i against an observer marking at most k vertices per step.

We describe a connected k-strategy that allows the observer to win in T when it can mark $f_k(v_0)$ extra vertices during its first step. First, the observer marks all nodes in $J = (N[v_0] \cup (\bigcup_{1 \leqslant i \leqslant r} I_i)) \setminus \{v_0\}$. The set $J \cup \{v_0\}$ induces a connected subtree of T and it is possible since $|J| \leqslant f_k(v_0) + k$. Then the fugitive moves to some child v_i $(1 \leqslant i \leqslant r)$ of v_0. Since the vertices of I_i and v_i are already marked, the observer will win in T_i in a connected way.

On the contrary, if strictly less than $f_k(v_0)$ vertices can be marked initially in $V(T) \setminus \{v_0\}$, then there is at least one child v_i $(1 \leqslant i \leqslant r)$ such that either v_i is not marked after the first step of the observer, or at most $f_k(v_i) - 1$ vertices in $V(T_i) \setminus \{v_i\}$ are marked after the first step of the observer. In both cases, the fugitive will win in T_i. □

Theorem 17. *For any tree T and any v_0, the value of $sn(T, v_0) = csn(T, v_0)$ can be computed in time $O(n \cdot \log n)$.*

Proof. By definition of the connected variant of the surveillance game, we have $csn(T, v_0) \geqslant sn(T, v_0)$. The strategy for the observer described in the proof of Lemma 16 is connected. Thus $sn(T, v_0) = csn(T, v_0) = \min\{k\colon\ f_k(v_0) = 0\}$. Note that, v being fixed, $f_k(v)$ is a decreasing function of k. The result comes from the fact that $f_k(v_0)$ can be computed in linear time and so, the minimum k such that $f_k(v_0) = 0$ can be searched using dichotomy. □

We now give a combinatorial characterization of $sn(T, v_0)$ for any tree T rooted at v_0.

Lemma 18. *For any tree T, for any $v_0 \in V(T)$, and for any $k < sn(T, v_0)$, there is a set of vertices $S \subseteq V(T)$ inducing a subtree of T containing v_0 and such that $\lceil \frac{|N[S]|-1}{|S|} \rceil > k$.*

Proof. Let $k < sn(T, v_0)$. By Lemma 16, $f_k(v_0) > 0$. Let S be the inclusion-maximal subtree of T containing v_0 and such that $f_k(v) > 0$ for all vertices v in S. We show by induction on the height of S that $f_k(v_0) = |N[S]| - 1 - k|S|$. If $S = \{v_0\}$ and v_0 has degree d, then $f_k(v_0) = d - k = |N[S]| - 1 - k|S| > 0$ because for any child v of v_0, $f_k(v) = 0$.

Assume that the result holds for any subtree of height at most h and assume that S has height $h+1$. Let d be the degree of v_0 and let $v_1, \ldots, v_r$, $1 \leqslant r \leqslant d$, be the children of v_0 with $f_k(v_i) > 0$. Let S_i be the subtree of S rooted at v_i, $1 \leqslant i \leqslant r$, and let $N[S_i]$ be the vertices of S_i or in the neighborhood of S_i in the subtree of T rooted at v_i. By the induction hypothesis, $f_k(v_i) = |N[S_i]| - 1 - k|S_i|$ for all $1 \leqslant i \leqslant r$. Now, $f_k(v_0) = d - k + \sum_{1 \leqslant i \leqslant r} f_k(v_i) = d - k + \sum_{1 \leqslant i \leqslant r}(|N[S_i]| - 1 - k|S_i|) = d - k + (|N[S]| - 1 - (d - r)) - r - k(|S| - 1) = |N[S]| - 1 - k|S|$. □

Lemma 19. *For any tree T, for any $v_0 \in V(T)$, for any $k \geqslant sn(T, v_0)$, for any $S \subseteq V(T)$ inducing a subtree of T containing v_0, we have $\lceil \frac{|N[S]|-1}{|S|} \rceil \leqslant k$.*

Proof. We consider the following game. Initially, an unbounded number of fugitives are in v_0 which is initially marked. Then, at most k vertices of $T \setminus \{v_0\}$ are marked. At each turn, each fugitive can move along an edge of the tree, and then, for each vertex v that is reached for the first time by a fugitive, at most k vertices can be marked in T_v the subtree of T rooted at v. The fugitives win if at least one fugitive reaches an unmarked vertex. They lose otherwise.

We first show that if $k \geqslant sn(T, v_0)$ then the fugitives lose in this game. Assume that $k \geqslant sn(T, v_0)$. Then there is a winning k-strategy σ for the "normal" surveillance game in T starting from v_0. Recall that by Theorem 7, we can restrict the fugitive to follow an induced path. Since for any $t \in V(T)$, there is a unique induced path from v_0 to t, σ can be defined uniquely by the position of the fugitive. That is, in the case of trees, we can define a k-strategy as a function that assigns a subset $\sigma(t) \subseteq V(T_t)$ (of size at most k) to any vertex $t \in V(T)$. Now, in the game with several fugitives, we consider the following strategy: each time a vertex t is reached for the first time by a fugitive, we mark the vertices in $\sigma(t)$. The fugitives cannot win against such a strategy.

Finally, we show that if there is a subtree S containing v_0 such that $\lceil \frac{|N[S]|-1}{|S|} \rceil > k$, then the fugitives win the new game. Indeed, the fugitives first occupy all vertices of S. At this step, at most $k \cdot |S| + 1$ vertices have been marked (because S is connected and v_0 is marked and for each vertex in S at most k vertices in $V(T) \setminus \{v_0\}$ are marked). Since $|N[S]| > k \cdot |S| + 1$, at least one unmarked vertex in $N[S]$ will be reached by some fugitive during the next step.

Hence, $sn(T, v_0) \geqslant \max\lceil \frac{|N[S]|-1}{|S|} \rceil$, where the maximum is taken over all $S \subseteq V(T)$ inducing a subtree of T containing v_0. □

Theorem 20. *For any graph G and $v_0 \in V(G)$, we have $sn(G, v_0) \geqslant \max\lceil \frac{|N[S]|-1}{|S|} \rceil$, where the maximum is taken over all subsets $S \subseteq V(G)$ inducing a connected subgraph of G containing v_0. Moreover, there is equality in the case of trees.*

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

Proof. Let $S \subseteq V(G)$ that induces a connected subgraph containing v_0. Let T_S be a spanning tree of $G[S]$ rooted at v_0 and let T be a spanning tree of G having T_S as a subtree. Clearly, $sn(G, v_0) \geqslant sn(T, v_0)$. By Lemma 19, $sn(T, v_0) \geqslant \lceil \frac{|N[S]|-1}{|S|} \rceil$. Hence, $sn(G, v_0) \geqslant \max \lceil \frac{|N[S]|-1}{|S|} \rceil$, the maximum being taken over all $S \subseteq V(G)$ inducing a connected subgraph of G containing v_0.

In the case of trees, the equality follows from Lemma 18. □

4.2. Keeping interval graphs under surveillance

We recall that an *interval graph* G is the intersection graph of a set of real intervals. This set of real intervals is a *realization* of G. In this section, we give a polynomial-time algorithm for computing the surveillance number in interval graphs. Moreover, we show that surveillance numbers for the connected variant and the unrestricted variant are equal in this class of graphs. It is important to recall that, by Theorems 7 and 8, we do not help the observer when forcing the fugitive to move at each step and to follow induced paths (in both variants). Hence, in this section, we assume that the fugitive obeys these restrictions.

Let G be a connected interval graph and $v_0 \in V(G)$. We consider any realization $\mathcal{I}$ of G such that, no two intervals have a common end (such a realization always exists). Let us say that $v \prec_L w$ if the left (smallest) end of (the interval of) v is smaller than the left end of w, and $v \succ_R w$ if the right (largest) end of v is larger than the right end of w.

We partition $V(G)$ into several subsets. Let $\mathcal{C}$ be the subset of vertices the interval of which contains the interval of v_0. Note that $v_0 \in \mathcal{C} \subseteq N[v_0]$ and that $\mathcal{C}$ induces a clique. Since G is an interval graph, $V \setminus N[v_0]$ induces a subgraph H the connected components of which are interval graphs. Let $\mathcal{L}$ be the vertices of the components of H with their interval more to the left than the interval of v_0, i.e., the largest end of an interval in $\mathcal{L}$ is strictly smaller than the smallest end of the interval of v_0. Similarly, let $\mathcal{R}$ be the vertices of the components of H with their interval more to the right than the interval of v_0. Note that $\mathcal{R}$ and $\mathcal{L}$ are disjoint and are separated by $N[v_0]$ because G is an interval graph: no interval can be both more to the left and more to the right than the interval of v_0. Let $\mathcal{C}_L$ be the vertices in $N(v_0) \setminus \mathcal{C}$ with neighbors in $\mathcal{L}$ and let $\mathcal{C}_R$ be the vertices in $N(v_0) \setminus \mathcal{C}$ with neighbors in $\mathcal{R}$. Finally, let $\mathcal{C}' = V(G) \setminus (\mathcal{L} \cup \mathcal{C}_L \cup \mathcal{C} \cup \mathcal{C}_R \cup \mathcal{R})$. Note that, for any $v \in \mathcal{C}'$, $v_0 \in N(v) \subseteq N[v_0]$.

Claim 21. *$(\mathcal{L}, \mathcal{C}_L, \mathcal{C}, \mathcal{C}', \mathcal{C}_R, \mathcal{R})$ is a partition of $V(G)$.*

Proof. Since, $N[v_0] = \mathcal{C} \cup \mathcal{C}' \cup \mathcal{C}_R \cup \mathcal{C}_L$, $(\mathcal{L}, \mathcal{C}_L, \mathcal{C}, \mathcal{C}', \mathcal{C}_R, \mathcal{R})$ covers $V(G)$. It only remains to prove that $\mathcal{C}_R \cap \mathcal{C}_L = \emptyset$. Indeed, if $v \in \mathcal{C}_R \cap \mathcal{C}_L$, then v must be adjacent to a vertex in $\mathcal{L}$ and to a vertex in $\mathcal{R}$. However, it means that the interval of v contains the one of v_0 and $v \in \mathcal{C}$, a contradiction. □

Recall that the fugitive is forced to follow an induced path. We now describe the structure of induced paths in G. Roughly, the next lemma says that once the fugitive has chosen a "side" (left or right) it has to remain on this side, and the choice occurs after one or two moves. Moreover, once the fugitive has chosen a side, it must go "further" into this side or it should stop.

Lemma 22. *Let $P = (v_0, v_1, \ldots, v_p)$ be an induced path starting from v_0 in any connected interval graph G. Let $\mathcal{L}, \mathcal{C}_L, \mathcal{C}, \mathcal{C}', \mathcal{C}_R, \mathcal{R}$ be defined as above. Then, there are three possible cases*:

1. *Either $v_1 \in \mathcal{C}'$ and then $p = 1$;*
2. *Either $v_2 \in \mathcal{L}$, and*
   - *for all $i > 1$, $v_i \in \mathcal{L}$ and $v_1 \in \mathcal{C}_L \cup \mathcal{C}$;*
   - *for all $0 < i < p - 1$, $v_{i+1} \prec_L v_i$;*
   - *if $v_{p-1} \prec_L v_p$ then $N(v_p) \subseteq N(v_{p-1})$.*
3. *Or $v_2 \in \mathcal{R}$, and*
   - *for all $i > 1$, $v_i \in \mathcal{R}$ and $v_1 \in \mathcal{C}_R \cup \mathcal{C}$;*
   - *for all $0 < i < p - 1$, $v_{i+1} \succ_R v_i$;*
   - *if $v_{p-1} \succ_R v_p$ then $N(v_p) \subseteq N(v_{p-1})$.*

Proof. Clearly, v_2 cannot be in $N[v_0] = \mathcal{C} \cup \mathcal{C}' \cup \mathcal{C}_R \cup \mathcal{C}_L$ because P is induced. Hence, $v_2 \in \mathcal{R} \cup \mathcal{L}$. If $v_1 \in \mathcal{C}'$, then all neighbors of v_1 are in $N[v_0]$ and thus $p = 1$. Let us assume that $v_2 \in \mathcal{L}$. The case $v_2 \in \mathcal{R}$ can be dealt with similarly.

By the previous remark, v_2 cannot be adjacent to a vertex in $\mathcal{C}_R \cup \mathcal{C}'$. Hence, $v_1 \in \mathcal{C}_L \cup \mathcal{C} = N[v_0] \setminus (\mathcal{C}_R \cup \mathcal{C}')$. Moreover, for all $i > 1$, $v_i \notin N[v_0]$ because P is induced. Since $N[v_0]$ separates $\mathcal{R}$ and $\mathcal{L}$, and since $v_2 \in \mathcal{L}$, for all $i > 1$, $v_i \in \mathcal{L}$.

Let us assume that $v_i \prec_L v_{i+1}$ for some $0 < i < p$ such that i is minimum with this property. We show that $i = p - 1$ and $N(v_p) \setminus \bigcup_{j<p} N(v_j) = \emptyset$.

We first consider the case when the interval of v_i does not contain the interval of v_{i+1}. Since $v_i \prec_L v_{i+1}$, we get that $v_{i+1} \succ_R v_i$. Since P is induced, $v_{i+1} \notin N(v_{i-1})$. This implies, since $v_i \in N(v_{i-1})$, that $v_i \succ_R v_{i-1}$. Since, by minimality of i, $v_i \prec_L v_{i-1}$, the interval of v_{i-1} must be contained in the interval of v_i. Therefore, if $i > 1$, then $N(v_{i-1}) \subseteq N(v_i)$, which

contradicts $v_{i-2} \in N(v_{i-1}) \setminus N(v_i)$. Then $i = 1$ and the interval of $v_{i-1} = v_0$ is strictly more to the left than the interval of $v_{i+1} = v_2$ which contradicts the fact that $v_2 \in \mathcal{L}$.

Therefore, the interval of v_{i+1} must be contained into the interval of v_i. Then $N(v_{i+1}) \subseteq N(v_i) \subseteq \bigcup_{j \leqslant i} N(v_j)$. If $i < p-1$, then $v_{i+2} \in N(v_{i+1}) \cap N(v_i)$ contradicting the fact that P is induced. Hence $i = p-1$, and we get $N(v_p) \setminus \bigcup_{j<p} N(v_j) = \emptyset$. □

The next lemma shows that, in interval graphs, we can define few particular induced paths that "dominate" all paths. That is, if the observer is able to win when the fugitive is constrained to follow one of these particular paths, then the observer always wins.

Let $v_L \in \mathcal{C}_L$ be the smallest vertex of $\mathcal{C}_L$ according to $\prec_L$, i.e., $v_L \prec_L w$ for every $w \in \mathcal{C}_L \setminus \{v_L\}$. For any $v_1 \in \mathcal{C} \cup \{v_L\}$, let us define $P_L(v_1)$ as the longest induced path $(v_0, v_1, v_2, \ldots, v_p)$ such that, for all $i \geqslant 1$, v_{i+1} is the smallest vertex of $N(v_i)$ according to $\prec_L$, i.e., $v_{i+1} \prec_L w$ for every $w \in N(v_i)$. Intuitively, except for the first move, we choose as next vertex the neighbor with leftmost left end.

Symmetrically, let $v_R \in \mathcal{C}_R$ be the largest vertex of $\mathcal{C}_R$ according to $\succ_R$, i.e., $v_R \succ_R w$ for any $w \in \mathcal{C}_R \setminus \{v_R\}$. For any $v_1 \in \mathcal{C} \cup \{v_R\}$, let us define $P_R(v_1)$ as the longest induced path $(v_0, v_1, v_2, \ldots, v_p)$ such that, for all $i \geqslant 1$, v_{i+1} is the largest vertex of $N(v_i)$ according to $\succ_R$, i.e., $v_{i+1} \succ_R w$ for every $w \in N(v_i)$. Except for the first move, we choose as next vertex the neighbor with rightmost right end.

Finally, for any path $P = (v_0, \ldots, v_p)$ and for all $i \leqslant p$, let $P^i = N[\{v_0, \ldots, v_i\}]$ the set of the vertices that are in $\{v_0, \ldots, v_i\}$ or that have a neighbor in $\{v_0, \ldots, v_i\}$.

Lemma 23. *Let G be a connected interval graph, let $P = (v_0, \ldots, v_p)$ be an induced path starting from v_0 with $p > 1$, and let Q be the path defined as follows*:

- $Q = P_L(v_L)$ *if* $v_1 \in \mathcal{C}_L$;
- $Q = P_R(v_R)$ *if* $v_1 \in \mathcal{C}_R$;
- $Q = P_L(v_1)$ *if* $v_1 \in \mathcal{C}$ *and* $v_2 \in \mathcal{L}$, *and*
- $Q = P_R(v_1)$ *if* $v_1 \in \mathcal{C}$ *and* $v_2 \in \mathcal{R}$.

For all $i \leqslant p$, we have $P^i \subseteq Q^{i'}$, where $i' = \min\{i, |Q| - 1\}$.

Proof. By Lemma 22, P must satisfy one of the four cases.

Let us first assume that $v_1 \in \mathcal{C}_L$, and let $Q = P_L(v_L) = (v_0, v_L, q_2, q_3, \ldots, q_{p'})$. By Lemma 22, $v_i \in \mathcal{L}$ for all $i \geqslant 2$. Hence, since no vertices of $\mathcal{L} \cup \mathcal{C}_L$ have a neighbor in $\mathcal{R}$, we have $N[P] \subseteq \mathcal{L} \cup N[v_0]$.

Similarly, $N[Q] \subseteq \mathcal{L} \cup N[v_0]$. We show that $N[Q] = \mathcal{L} \cup N[v_0]$. Clearly, $N[v_0] \subseteq N[Q]$. Let $v \in \mathcal{L} \setminus Q$. Note that $\mathcal{L} \cup N[v_0]$ is connected and let $R = (v_0, a_1, a_2, \ldots, a_i, v)$ $(i \geqslant 1)$ be a shortest path from v_0 to v. Since, $v \in \mathcal{L}$ and R is a shortest path, we get $a_i \prec_L a_{i-1} \prec_L \cdots \prec_L a_1 \prec_L v_0$. Furthermore, either $v \prec_L a_i$ or a_i contains v. By induction on $j \leqslant i$, we show that $(v_0, v_L, q_2, \ldots, q_j, a_{j+1}, \ldots, a_i, v)$ is a shortest path from v_0 to v and therefore $v \in N[Q]$. Since v_L is the smallest neighbor of v_0 according to $\prec_L$, then $a_2 \in N[v_L]$ (or $v \in N[v_L]$ if $i = 1$) and then $(v_0, v_L, a_2, \ldots, a_i, v)$ is a shortest path from v_0 to v. Assume that $(v_0, v_L, q_2, \ldots, q_j, a_{j+1}, \ldots, a_i, v)$ is a shortest path from v_0 to v for some $j < i$. Then, $j < p'$ because otherwise, Q would not be a maximal induced path. Moreover, since q_{j+1} is the smallest neighbor of q_j according to $\prec_L$, then $a_{j+2} \in N[q_{j+1}]$ (or $v \in N[q_{j+1}]$ if $j = i-1$) and then $(v_0, v_L, q_2, \ldots, q_j, q_{j+1}, a_{j+2}, \ldots, a_i, v)$ is a shortest path from v_0 to v. Hence, $\mathcal{L} \cup N[v_0] \subseteq N[Q]$ and so $N[Q] = \mathcal{L} \cup N[v_0]$.

Therefore, for all $i \geqslant p' = |Q| - 1$, $P^i \subseteq \mathcal{L} \cup N[v_0] = N[Q] = Q^{p'}$.

Now, we show by induction on $i < p'$ that $P^i \subseteq Q^i$. Since $P^0 = Q^0 = N[v_0]$, the result holds for $i = 0$. Let $0 \leqslant i < p' - 1$ and assume that $P^i \subseteq Q^i$. Let $v \in P^{i+1} \setminus P^i$. Then, $(v_0, v_1, \ldots, v_{i+1}, v)$ is an induced path from v_0 to v. As above, we show that $(v_0, v_L, q_2, \ldots, q_{i+1}, v)$ is an induced path from v_0 to v. Therefore, $v \in Q^{i+1}$ and the result holds.

The case $v_1 \in \mathcal{C}_R$ can be handled similarly by symmetry.

Now, if $v_1 \in \mathcal{C}$ and $v_2 \in \mathcal{L}$, we can prove in a similar way that $N[P] \subseteq \mathcal{L} \cup N[v_1] = N[P_L(v_1)]$. Hence, for all $i \geqslant p' = |P_L(v_1)| - 1$, $P^i \subseteq \mathcal{L} \cup N[v_0] = N[P_L(v_1)] = Q^{p'}$. Moreover, a similar induction on $i < p'$ allows to prove that $P^i \subseteq Q^i$. The case $v_1 \in \mathcal{C}$ and $v_2 \in \mathcal{R}$ is symmetric. □

The previous two lemmas roughly say that the fugitive can only choose five kinds of induced paths: the ones with second vertex in $\mathcal{C}'$ (such induced paths have only one edge), the ones with second vertex in $\mathcal{C}_L$ and going through $\mathcal{L}$, those with second vertex in $\mathcal{C}_R$ and going through $\mathcal{R}$, and the paths with second vertex $x \in \mathcal{C}$ and then either going through $\mathcal{L}$ or through $\mathcal{R}$. Moreover, once the observer knows which kind of path has been chosen, it is sufficient for it to protect one particular path. However, during the first step (before the first move of the fugitive), the first set S_0 of marked vertices must be chosen by the observer independently of what the fugitive will choose. Similarly, if the fugitive first goes to $x \in \mathcal{C}$, the observer cannot guess yet on which side the fugitive will flee. Hence, the set S_x of marked vertices during the second step (before the second move of the fugitive) must be independent of the next choice of the fugitive (S_x may only depend on x). The next theorem formalizes these ideas.

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

Now, we order the vertices of $V(G)$ in the following way. The vertices in $N[v_0]$ are ordered arbitrarily, any vertex in $\mathcal{L}$ is smaller than a vertex in $N[v_0]$ and any vertex in $\mathcal{R}$ is larger than the vertices in $N[v_0]$. Finally, vertices in $\mathcal{L}$ are ordered according to $\succ_R$, i.e., for any $v, w \in \mathcal{L}$, $v < w$ if, and only if, $w \succ_R v$. Symmetrically, vertices in $\mathcal{R}$ are ordered according to $\prec_L$. We say that a set $S \subseteq V(G)$ is *contiguous* if for any $a, b \in S$ and $a < w < b$, then $w \in S$. Note that a contiguous subset including $N[v_0]$ of a connected interval graph induces a connected subgraph.

Theorem 24. *Let G be an interval graph and $v_0 \in V(G)$. Let k be the smallest integer such that*:

- *there exists $S_0 \subseteq V(G)$ with $N[v_0] \subseteq S_0$, $|S_0 \setminus \{v_0\}| \leqslant k$, and*
- *for all $i > 0$, $|P_L^i(v_L) \setminus S_0| \leqslant i \cdot k$ and $|P_R^i(v_R) \setminus S_0| \leqslant i \cdot k$, and*
- *for any $x \in \mathcal{C} \setminus \{v_0\}$, there is $S_x \subseteq V(G) \setminus S_0$ with $N[v_0] \cup N[x] \subseteq S_0 \cup S_x$ and $|S_x| \leqslant k$, and*
- *for all $i > 1$ and any $x \in \mathcal{C} \setminus \{v_0\}$, $|P_L^i(x) \setminus (S_0 \cup S_x)| \leqslant (i-1) \cdot k$ and $|P_R^i(x) \setminus (S_0 \cup S_x)| \leqslant (i-1) \cdot k$.*

Then, $sn(G, v_0) = csn(G, v_0) = k$. Moreover, S_0 can be chosen contiguous and the sets S_x can be chosen such that $S_0 \cup S_x$ is contiguous for any $x \in \mathcal{C}$ (without increasing k).

Proof. Let k be the smallest integer defined as in the statement of the theorem. We first show that $k' = sn(G, v_0)$ is at least k.

Claim 25. $sn(G, v_0) = k' \geqslant k$.

Let σ be any optimal winning strategy for the observer, i.e., marking k' vertices at each step. Let $S_0 = \sigma(\{v_0\}, v_0) \cup \{v_0\}$ and, for any $x \in \mathcal{C}$, let $S_x = \sigma(S_0, x)$. Obviously, $|S_0| \leqslant k' + 1$, $|S_x| \leqslant k'$, $N[v_0] \subseteq S_0$ and $N[x] \cup N[v_0] \subseteq S_0 \cup S_x$.

Now, assume that the fugitive follows the induced path $P_L(x) = (v_0, x, v_2, \ldots, v_p)$ for some $x \in \mathcal{C}$. At step i ($1 < i \leqslant p$), when it is the turn of the fugitive that stands at v_i, the observer must have marked at least the vertices in $N[\{v_0, x, v_2, \ldots, v_i\}]$. Moreover, during the first two steps of the strategy, the observer has marked the vertices of S_0 and S_x by definition of σ. Hence, during the $i-1$ steps after the first two steps, the observer must have marked the vertices in $N[\{v_0, x, v_2, \ldots, v_i\}] \setminus (S_0 \cup S_x) = P_L^i(x) \setminus (S_0 \cup S_x)$ which proves that $|P_L^i(x) \setminus (S_0 \cup S_x)| \leqslant (i-1) \cdot k'$.

The other properties can be proved in the same way and thus, $k' \geqslant k$.

Claim 26. *There exist S_0^* and S_x^* ($x \in \mathcal{C}$) that are contiguous sets and that satisfy the same properties as S_0 and S_x ($x \in \mathcal{C}$). In other words, S_0 and S_x ($x \in \mathcal{C}$) may be chosen contiguous.*

Proof. Recall that to define contiguous sets, we have ordered the vertices in $V(G)$.

Let $\ell = |S_0 \cap \mathcal{L}|$ and $r = |S_0 \cap \mathcal{R}|$. Let S_0^* be the set obtained from the union of $N[v_0]$, the ℓ greatest vertices in $\mathcal{L}$ and the r smallest vertices in $\mathcal{R}$. Note that S_0^* is contiguous and that $S_0 = S_0^*$ if, and only if, S_0 is contiguous.

Similarly, for any $x \in \mathcal{C}$, let $\ell_x = |S_x \cap \mathcal{L}|$ and $r_x = |S_x \cap \mathcal{R}|$. Let S_x^* be the set obtained from the union of the ℓ_x greatest vertices in $\mathcal{L} \setminus S_0^*$ and the r_x smallest vertices in $\mathcal{R} \setminus S_0^*$. Note that $S_0^* \cup S_x^*$ is contiguous and that $S_x = S_x^*$ if, and only if, $S_0 \cup S_x$ is contiguous.

We claim that S_0^* and the sets S_x^*, $x \in \mathcal{C}$, satisfy the desired properties.

Indeed, $(N[v_0], S_0 \cap \mathcal{L}, S_0 \cap \mathcal{R})$ is a partition of S_0 hence, $k + 1 = |N[v_0]| + r + \ell$ and then $|S_0^*| = k + 1$ and $N[v_0] \subseteq S_0^*$. Moreover, $|S_x^*| = \ell_x + r_x = |S_x| \leqslant k$. Since $N[v_0] \cup N[x] \subseteq S_0 \cup S_x$, $N[x] \cap \mathcal{L} \subseteq (S_0 \cup S_x) \cap \mathcal{L}$. Hence, $|N[x] \cap \mathcal{L}| \leqslant \ell + \ell_x$. Moreover, $N[x] \cap \mathcal{L}$ must be contiguous. Therefore, $N[x] \cap \mathcal{L} \subseteq (S_0^* \cup S_x^*) \cap \mathcal{L}$. By symmetry, $N[x] \cap \mathcal{R} \subseteq (S_0^* \cup S_x^*) \cap \mathcal{R}$, and then $N[v_0] \cup N[x] \subseteq S_0^* \cup S_x^*$.

Let $i > 1$ and $x \in \mathcal{C} \setminus \{v_0\}$. We have $|P_L^i(x) \setminus (S_0 \cup S_x)| \leqslant (i-1) \cdot k$. Moreover, $P_L^i(x) \setminus (S_0 \cup S_x) \subseteq \mathcal{L}$, hence $|P_L^i(x) \cap \mathcal{L}| \leqslant (i-1) \cdot k + \ell + \ell_x$. Also, $P_L^i(x) \cap \mathcal{L}$ must be contiguous, so either $P_L^i(x) \subseteq S_0^* \cup S_x^*$ in which case the result is trivial, or $(S_0^* \cup S_x^*) \cap \mathcal{L} \subseteq P_L^i(x) \cap \mathcal{L}$. In the latter case, $|P_L^i(x) \setminus (S_0^* \cup S_x^*)| = |P_L^i(x) \cap \mathcal{L}| - |(S_0^* \cup S_x^*) \cap \mathcal{L}| \leqslant (i-1) \cdot k$.

The other properties can be checked in a similar way. □

Now, we describe a k-winning connected strategy for the observer. According to Claim 26, we may assume that the sets S_0 and S_x ($x \in \mathcal{C}$) are contiguous.

Claim 27. $csn(G, v_0) \leqslant k$.

Proof. We define a winning, connected k-strategy for the observer when the fugitive is constrained to follow induced paths. By Theorem 7, this is sufficient to prove the claim.

Initially, the observer marks the vertices in S_0. Let $P = (v_0, v_1, \ldots)$ be an induced path followed by the fugitive starting from v_0. If $v_1 \in \mathcal{C}'$ then the fugitive must stop there and loses. So assume that $v_1 \notin \mathcal{C}'$. By Lemma 22, $v_1 \in \mathcal{C} \cup \mathcal{C}_R \cup \mathcal{C}_L$ and $v_2 \in \mathcal{L} \cup \mathcal{R}$. First: if $v_1 \in \mathcal{C}$, at the second step, the observer marks the vertices in S_{v_1}. Next, if $v_1 \in \mathcal{C}_L$ or $v_2 \in \mathcal{L}$, then at

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

each step (but the second one if $v_1 \in \mathcal{C}$), the observer marks the k greatest vertices unmarked in $\mathcal{L}$. Finally, if $v_1 \in \mathcal{C}_R$ or $v_2 \in \mathcal{R}$, then at each step (but the second one if $v_1 \in \mathcal{C}$), the observer marks the k smallest vertices unmarked in $\mathcal{R}$. Such a strategy is connected since, at each step, the set of marked vertices is contiguous and connected to the set of previously marked vertices. It is a k-strategy: the observer marks at most k vertices at each step because $|S_0 \setminus \{v_0\}| \leqslant k$ and $|S_{v_1}| \leqslant k$.

Let us show that the strategy is winning. Assume that $v_1 \in \mathcal{C}$ and $v_2 \in \mathcal{L}$; the other cases can be handled in a similar way. Clearly, the fugitive cannot win during the first two steps since $N[v_0] \subseteq S_0$ and $N[v_0] \cup N[v_1] \subseteq S_0 \cup S_{v_1}$. Now, after its i-th step, $i > 2$, the observer has marked the vertices in $S_0 \cup S_{v_1}$ and the vertices in the set M formed with the $(i-2) \cdot k$ greatest vertices in $\mathcal{L} \setminus (S_0 \cup S_{v_1})$. Since M is contiguous, $P_L^{i-1}(v_1) \setminus (S_0 \cup S_{v_1})$ is also contiguous and $|P_L^{i-1}(v_1) \setminus (S_0 \cup S_{v_1})| \leqslant (i-2) \cdot k$, we get that $P_L^{i-1}(v_1) \setminus (S_0 \cup S_{v_1}) \subseteq M$. Finally, by Lemma 23, $P^{i-1} = N[\{v_0, v_1, \ldots, v_{i-1}\}] \subseteq P_L^{i-1}(v_1)$. Therefore, $P^{i-1} \subseteq M \cup (S_0 \cup S_{v_1})$. Hence, all neighbors of the current position v_{i-1} of the fugitive are marked and the fugitive cannot escape during its next move.

Hence, $csn(G, v_0) \leqslant k$. □

This concludes the proof of Theorem 24. □

Theorem 28. *Given an n-node interval graph with maximum degree Δ and a node $v_0 \in V(G)$, the value of $sn(G, v_0)$ (respectively, $csn(G, v_0)$) can be computed in time $O(n \cdot \Delta^3)$.*

Proof. By Theorem 24, it is sufficient to prove that the smallest integer k defined in Theorem 24 can be computed in polynomial time. An exhaustive check is sufficient: k being fixed, it can be checked in polynomial-time whether k satisfies the properties. Indeed, by Theorem 24, the sets S_0 and S_x ($x \in \mathcal{C}$) that must be checked can be restricted to be contiguous.

Consequently, since S_0 has $k+1$ vertices and must contain v_0, there are at most k such sets. Then, for any $x \in \mathcal{C}$, S_0 being fixed, there are at most k sets S_x since $S_0 \cup S_x$ must be contiguous. Moreover, given $x, x' \in \mathcal{C}$, S_x and $S_{x'}$ can be checked independently.

Hence, for any integer k, we have to check at most k sets S_0 and k sets S_x for each $x \in \mathcal{C}$. Since each test can be done in linear time with respect to n, since $\mathcal{C} \subseteq N[v_0]$ and $k \leqslant \Delta$, the complexity of the algorithm is $O(n \cdot \Delta^3)$. □

5. Conclusion and further work

In this section, we summarize open questions and we discuss the different variants we plan to investigate.

We first solve the case of an invisible fugitive. Generally, in cops and robber games, both visible and invisible robbers are difficult to handle. In our game, the invisible (or blind) case is trivial. In the case of an invisible fugitive, a winning k-strategy for the observer is a sequence $(X_1, \ldots, X_r)$ of subsets of vertices of G such that $|X_i| \leqslant k$ for all $i \leqslant r$ and for any walk W (followed by the fugitive) starting from v_0 and of length i, $W \subseteq \bigcup_{j \leqslant i} X_j \cup \{v_0\}$. The strategy is connected if $\bigcup_{j \leqslant i} X_j \cup \{v_0\}$ induces a connected subgraph for all $i \leqslant r$. Let $bsn(G, v_0)$ ($cbsn(G, v_0)$) be the smallest k such that there exists a (connected) winning k-strategy for the observer. It is straightforward that:

Theorem 29. *For any connected (di)graph G and $v_0 \in V(G)$, $bsn(G, v_0) = cbsn(G, v_0)$ and equals the smallest k such that, for all $i \geqslant 1$, $|V_i| \leqslant ki + 1$, where V_i is the set of vertices at distance at most i from v_0. Moreover, it can be computed in linear time in the number of edges.*

We now recall the questions we have asked throughout the paper and add some new questions:

- Does the problem of deciding *sn* in undirected graphs belong to NP?
- Does there exist a constant bounding the ratio (or the difference) between *csn* and *sn* in any graph?
- Does there exist a constant bounding the ratio (or the difference) between $\max \lceil \frac{|N[S]|-1}{|S|} \rceil$, where the maximum is taken over all $S \subseteq V(G)$ inducing a connected subgraph of G containing v_0, and $sn(G, v_0)$ for any graph G and any v_0?
- What is the complexity of computing the surveillance number in the class of graphs with maximum degree 4? With bounded degree? With bounded treewidth?
- Do there exist a constant $c < 4$ and an algorithm that computes $sn(G, v_0)$ in time $O(c^n)$ in general graphs G?

To conclude, we discuss the different variants of the problem we plan to study in the future. In this paper, the strongest assumption is probably about the unbounded memory: when a Web page is prefetched, then it remains prefetched. In other words, a vertex that is marked remains marked for all the following steps of the surveillance game. We plan to investigate two more realistic models corresponding to two cache management policies. The first variant assumes that a marked vertex becomes unmarked after a constant number of steps. The second model allows the observer to unmark some vertices, respecting the constraint that the total number of nodes that are marked never exceeds a given threshold corresponding to the maximum number of Web pages that can be prefetched simultaneously.

We also plan to model the variant in which the minimum visiting time can be different among the pages.

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009

Finally, we believe that the connected version of the game is particularly interesting since it is closer to the more realistic *online* version of the prefetching problem. In an online version, the observer has no global knowledge of the graph but discovers progressively the neighbors of the vertices he marks. We will further investigate this variant.

Acknowledgement

We would like to thank the anonymous reviewers for the thorough reading of the paper and helpful comments.

References

[1] M. Aigner, M. Fromme, A game of cops and robbers, Discrete Appl. Math. 8 (1984) 1–12.
[2] N. Alon, A. Mehrabian, On a generalization of Meyniel's conjecture on the cops and robbers game, Electron. J. Comb. 18 (1) (2011).
[3] B. Alspach, Searching and sweeping graphs: a brief survey, Le Matematiche (2004) 5–37.
[4] T. Andreae, On a pursuit game played on graphs for which a minor is excluded, J. Comb. Theory, Ser. B 41 (1) (1986) 37–47.
[5] B. Bollobas, G. Kun, I. Leader, Cops and robbers in a random graph, J. Comb. Theory, Ser. B 103 (2) (2013) 226–236.
[6] A. Bonato, E. Chiniforooshan, P. Pralat, Cops and robbers from a distance, Theor. Comput. Sci. 411 (43) (2010) 3834–3844.
[7] J. Chalopin, V. Chepoi, N. Nisse, Y. Vaxès, Cop and robber games when the robber can hide and ride, SIAM J. Discrete Math. 25 (1) (2011) 333–359.
[8] J. Cham, http://www.phdcomics.com/comics/archive.php?comicid=1456, 2012.
[9] F. Fomin, D.M. Thilikos, An annotated bibliography on guaranteed graph searching, Theor. Comput. Sci. 399 (3) (2008) 236–245.
[10] F.V. Fomin, S. Gaspers, P.A. Golovach, D. Kratsch, S. Saurabh, Parameterized algorithm for eternal vertex cover, Inf. Process. Lett. 110 (16) (2010) 702–706.
[11] F.V. Fomin, F. Giroire, A. Jean-Marie, D. Mazauric, N. Nisse, To satisfy impatient web surfers is hard, in: 6th International Conference on FUN with Algorithms (FUN), in: Lect. Notes Comput. Sci., vol. 7288, Springer, 2012, pp. 166–176, http://hal.inria.fr/inria-00625703/en/.
[12] F.V. Fomin, P.A. Golovach, A. Hall, M. Mihalák, E. Vicari, P. Widmayer, How to guard a graph?, in: 19th International Symposium on Algorithms and Computation (ISAAC), in: Lect. Notes Comput. Sci., vol. 5369, Springer, 2008, pp. 318–329.
[13] F.V. Fomin, P.A. Golovach, J. Kratochvíl, N. Nisse, K. Suchan, Pursuing a fast robber on a graph, Theor. Comput. Sci. 411 (7–9) (2010) 1167–1181.
[14] F.V. Fomin, P.A. Golovach, D. Lokshtanov, Guard games on graphs: Keep the intruder out!, in: 7th International Workshop on Approximation and Online Algorithms (WAOA), in: Lect. Notes Comput. Sci., vol. 5893, Springer, 2009, pp. 147–158.
[15] F.V. Fomin, P.A. Golovach, D. Lokshtanov, Cops and robber game without recharging, Theory Comput. Syst. 50 (4) (2012) 611–620.
[16] P. Frankl, Cops and robbers in graphs with large girth and Cayley graphs, Discrete Appl. Math. 17 (1987) 301–305.
[17] 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, 1990.
[18] A.S. Goldstein, E.M. Reingold, The complexity of pursuit on a graph, Theor. Comput. Sci. 143 (1) (1995) 93–112.
[19] J.L. Goldwasser, W. Klostermeyer, Tight bounds for eternal dominating sets in graphs, Discrete Math. 308 (12) (2008) 2589–2593.
[20] R. Grigoras, V. Charvillat, M. Douze, Optimizing hypervideo navigation using a Markov decision process approach, in: ACM Multimedia, 2002, pp. 39–48.
[21] D. Joseph, D. Grunwald, Prefetching using Markov predictors, in: ISCA, 1997, pp. 252–263.
[22] M. Mamino, On the computational complexity of a game of cops and robbers, Theor. Comput. Sci. 477 (2013) 48–56.
[23] A. Mehrabian, Lower bounds for the cop number when the robber is fast, Comb. Probab. Comput. 20 (4) (2011) 617–621.
[24] O. Morad, A. Jean-Marie, Optimisation en temps-réel du téléchargement de vidéos, in: Proc. of the 11th Congress of the French Operations Research Soc., 2010.
[25] R.J. Nowakowski, P. Winkler, Vertex-to-vertex pursuit in a graph, Discrete Math. 43 (1983) 235–239.
[26] A. Quilliot, Problèmes de jeux, de point fixe, de connectivité et de représentation sur des graphes, des ensembles ordonnés et des hypergraphes, Thèse de doctorat d'état, Université de Paris VI, France, 1983.
[27] W.J. Savitch, Relationships between nondeterministic and deterministic tape complexities, J. Comput. Syst. Sci. 4 (2) (1970) 177–192.
[28] B.S.W. Schröder, The copnumber of a graph is bounded by $\lfloor \frac{3}{2} genus(G) \rfloor + 3$, in: Categorical Perspectives, Kent, OH, 1998, in: Trends in Maths., 2001, pp. 243–263.
[29] Zona Research Inc., The economic impacts of unacceptable web-site download speeds, White paper, Redwood City, CA, http://www.webperf.net/info/wp_downloadspeed.pdf, 1999.

Please cite this article in press as: F.V. Fomin et al., To satisfy impatient Web surfers is hard, Theoretical Computer Science (2014), http://dx.doi.org/10.1016/j.tcs.2014.01.009