The mixed search game against an agile and visible fugitive is monotone

Guillaume Mescoff, Christophe Paul, Dimitrios M. Thilikos

The mixed search game against an agile and visible fugitive is monotone*

Guillaume Mescoff† Christophe Paul† Dimitrios M. Thilikos†

Abstract

We consider the mixed search game against an agile and visible fugitive. This is the variant of the classic fugitive search game on graphs where searchers may be placed to (or removed from) the vertices or slide along edges. Moreover, the fugitive resides on the edges of the graph and can move at any time along unguarded paths. The *mixed search number against an agile and visible fugitive* of a graph G, denoted $\mathsf{avms}(G)$, is the minimum number of searchers required to capture a fugitive in this graph searching variant. Our main result is that this graph searching variant is *monotone* in the sense that the number of searchers required for a successful search strategy does not increase if we restrict the search strategies to those that do not permit the fugitive to visit an already "clean" edge. This means that mixed search strategies against an agile and visible fugitive can be polynomially certified, and therefore that the problem of deciding, given a graph G and an integer k, whether $\mathsf{avms}(G) \leq k$ is in NP. Our proof is based on the introduction of the notion of *tight bramble*, that serves as an obstruction for the corresponding search parameter. Our results imply that for a graph G, $\mathsf{avms}(G)$ is equal to the Cartesian tree product number of G, that is, the minimum k for which G is a minor of the Cartesian product of a tree and a clique on k vertices.

Keywords: Graph searching game, bramble, Cartesian tree product number, tree decomposition.

1 Introduction

A *search game* on a graph is a two-player game opposing a fugitive and a group of searchers. The searchers win if they capture the fugitive, while the fugitive's objective is to avoid capture. The searchers and the fugitive play in turns. Search games on graphs were originally been introduced by Parsons [17] who defines the *edge search number* of a graph as the minimum number of searchers required to capture a fugitive moving along the edges. To capture the fugitive, a search program is seen as a sequence of searchers' moves. A move consists in either adding a searcher on a vertex, removing a searcher from a vertex or sliding a searcher along an edge. The fugitive is allowed to move along the edges of paths that are free of searchers. The *edge search number* is then the minimum number of searchers required to capture the fugitive (that is, the fugitive cannot escape through an unguarded path).

During the last decades, a large number of variants of seminal Parsons' edge search game have been studied. Each of these variants depends on the fugitive and searchers abilities. For example, the fugitive can be *visible* to the searchers or *invisible*. It can also be *lazy*, then it stays at its location as long it is not directly threatened by the searchers, or *agile*, then it may change its location at every round. Where the fugitive resides (on edges or on vertices), whether the searchers are allowed to slide on edges or not, how the fugitive is captured are other criteria used to define search game variants (see [8, 15] for surveys).

The theory of search games on graphs is strongly connected to the theory of width parameters and graph decompositions. It is well known that the node search number of a graph against an agile but visible fugitive corresponds to the *treewidth* of that graph [19]. The same equivalence also holds for the case of the node search number against a lazy but invisible fugitive [5]. The node search variant

*Research supported by projects DEMOGRAPH (ANR-16-CE40-0028), ESIGMA (ANR-17-CE23-0010), and the French-German Collaboration ANR/DFG Project UTMA (ANR-20-CE92-0027).

†LIRMM, Univ Montpellier, CNRS, Montpellier, France.

where the fugitive is agile and invisible corresponds to the *pathwidth* [12, 11, 6]. These three variants are usually defined as node search games as the searchers cannot slide along an edge and the fugitive is located on vertices rather than on edges. In the case of agile but visible fugitive (respectively lazy but invisible fugitive), the existence of an escape strategy for the fugitive can be derived from the existence of a *bramble* of large order [19] which is a certificate of large treewidth. Likewise, in the case of agile and invisible, a large *blockage* certifies a large pathwidth and allows an escape strategy [3].

Interestingly, a consequence of the equivalences discussed above between search numbers and width parameters is that the corresponding games can be proved to be *monotone*. Intuitively, a game is monotone if it allows to optimally search a graph without recontamination of the previously cleared part of the graph. Monotonicity is an important property as it implies that the problem of deciding the corresponding search number of a graph belongs to NP (see [8] for a discussion about monotonicity).

Our results. In this paper, we consider the so-called *mixed search game against an agile and visible fugitive*. In this variant, the fugitive resides on the edges and the searchers can capture the fugitive either by sliding along the edge location it resides or by occupying the two vertices incident to that edge location (see Subsection 2.5 for formal definitions). Up to our knowledge, the monotonicity issue of a mixed search game was only proved in the case of an agile and invisible fugitive [4]. We prove that the mixed search game against an agile and visible fugitive is monotone. To that aim, we first show that the corresponding search number of a graph is equal to the *Cartesian tree product number* of that graph [10], also known as the *largeur d'arborescence* [22]. To certify these two parameters, we introduce the notion of *loose tree-decomposition* which is a relaxation of the celebrated tree-decomposition associated to treewidth. This allows us to define the notion of *tight bramble* as the min-max counterpart of Cartesian tree product number. Then the monotonicity of the mixed search game against an agile and visible fugitive is obtained by proving that the existence of a tight bramble allows an escape strategy for the fugitive, while the existence of a losse tree-decomposition of a small width allows to derive a monotone search strategy of small cost.

2 Preliminairies

2.1 Basic concepts

In this paper, all graphs are finite, loopless, and without multiple edges. We denote by $V(G)$ the set of the vertices of G and by $E(G)$ the set of the edges of G. We use notation xy in to denote an edge $e = \{x, y\}$. Given a edge $xy \in E(G)$, we define $G - xy = (V(G), E(G) \setminus \{xy\})$. The graph resulting from the removal of a vertex subset S of $V(G)$ is denoted by $G - S$. If $S = \{v\}$ is a singleton we write $G - v$ instead of $G - \{v\}$. We define the subgraph of G *induced by* S as the graph $G[S] = G - (V(G) \setminus S)$. We say that S is *connected* (in G) if $G[S]$ is connected. A subset S of $V(G)$ is a *separator* of G if $G - S$ contains more connected components than G. We say that S separates $X \subseteq V(G)$ from $Y \subseteq V(G)$ if X and Y are subsets of distinct connected components of $G - S$.

Contracting an edge $e = xy$ in a graph G yields the graph $G_{/e}$ where $V(G_{/e}) = V(G) \setminus \{x, y\} \cup \{v_e\}$ and $E(G_{/e}) = \{uv \in E(G) \mid \{u, v\} \cap \{x, y\} = \emptyset\} \cup \{v_e v \mid xv \in E(G) \text{ or } yv \in E(G)\}$. A graph H is a *minor* of a graph G, which is denoted $H \preceq G$, if H can be obtained from G by a series of vertex or edge deletions and edge contractions. It is well known that if $H \preceq G$, then there exists a *minor model* of H in G, that is a function $\rho : V(H) \to 2^{V(G)}$ such that:

1. for every $x \in V(H)$, the set $\rho(x)$ is connected in G; and
2. for every distinct vertices $x, y \in V(H)$, $\rho(x) \cap \rho(y) = \emptyset$; and
3. for every edge $xy \in E(H)$, there exists an edge $x'y' \in E(G)$ with $x' \in \rho(x)$ and $y' \in \rho(y)$.

Given a set S, a *sequence of subsets of* S is defined as $\mathcal{S} = \langle S_1, \ldots, S_i, \ldots \rangle$ with $i \geq 1$, where for every $i \geq 1$, $S_i \subseteq S$. The symmetric difference between two sets A and B is denoted by $A \ominus B$.

A *pathway* $\mathcal{W}$ in a graph G is a sequence $\mathcal{W} = \langle e_1, e_2, \ldots, e_\ell \rangle$ of edges of G such that if $\ell > 1$, then for every $i \in [\ell - 1]$, the edges e_i and e_{i+1} are distinct and incident to a common vertex (that is,

$e_i \cap e_{i+1} = \{v\}$ with $v \in V(G)$). If $\mathcal{W}$ is a pathway starting at edge e and ending at edge e', we say that $\mathcal{W}$ is a (e, e')-pathway. Observe that a pathway may contains several occurence of the same edge.

2.2 Cartesian tree product number

Let T be a tree and k be a strictly positive integer. We use K_k in order to denote the complete graph on k vertices. We let $T^{(k)} = T \square K_k$ denote the *cartesian product* of T with K_k, that is: $T^{(k)}$ is the graph with vertex set $\{(x, i) \mid x \in V(T), i \in [k]\}$ and with an edge between (x, i) and (y, j) when $x = y$, or when $xy \in E(T)$ and $i = j$ (see Figure 1).

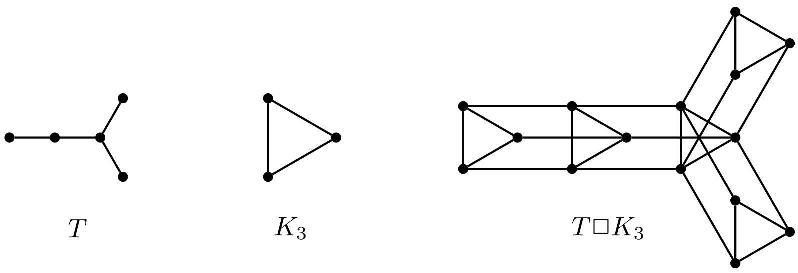

Figure 1: The cartesian product $T^{(3)} = T \square K_3$ of a tree T and the clique K_3.

Definition 1. *[10, 22] The* cartesian tree product number *of a graph G is*

$$\mathsf{ctp}(G) = \min\{k \in \mathbb{N} \mid G \preceq T^{(k)}\}.$$

Observe that for any tree T, as $T = T \square K_1$, $\mathsf{ctp}(T) = 1$. In [10], it is shown that the cartesian tree product number of a graph is equal to the *largeur d'arborescence* of that graph, a parameter introduced by Yves Colin De Verdière in [22].

2.3 Loose tree-decomposition

The concept of *loose tree-decomposition* is a relaxation of the celebrated *tree-decomposition* [9, 18] associated to the tree-width of a graph.

Definition 2 (Loose tree-decomposition)**.** *Let $G = (V, E)$ be a graph. A* loose tree-decomposition *is a pair $\mathcal{D} = (T, \chi)$ such that T is a tree and $\chi : V(T) \to 2^{V(G)}$ satisfying the following properties:*

(L1) *for every vertex $x \in V(G)$, the set $\{t \in V(T) \mid x \in \chi(t)\}$ induces a non-empty connected subgraph, say T_x, of T. We call T_x the* trace *of x in $\mathcal{D}$;*

(L2) *for every edge $e = xy \in E(G)$, there exists a tree-edge $f = \{t_1, t_2\} \in E(T)$ such that $e \in E(G[\chi(t_1) \cup \chi(t_2)])$;*

(L3) *for every tree-edge $f = \{t_1, t_2\}$, $\big|E(G[\chi(t_1) \cup \chi(t_2)]) \setminus \big(E(G[\chi(t_1)]) \cup E(G[\chi(t_2)])\big)\big| \leq 1$.*

We refer to the vertices of T as the nodes *and to their images via χ as the* bags *of the loose tree decomposition $\mathcal{D} = (T, \chi)$. The* width *of a loose tree-decomposition $\mathcal{D} = (T, \chi)$ of a graph G is defined as $\mathsf{width}(\mathcal{D}, G) = \max\{|\chi(t)| \mid t \in V(T)\}$.*

Figure 2 provides an example of a loose tree-decomposition. Hereafter, if $\mathcal{D} = (T, \chi)$ is a loose tree-decomposition of G, and $xy \in E(G[\chi(t_1) \cup \chi(t_2)]) \setminus (E(G[\chi(t_1)]) \cup E(G[\chi(t_2)]))$ for some adjacent nodes t_1 and t_2 (see condition (L3) above), then xy is called a *marginal edge*. We now examine important properties of loose tree-decompositions that will be used in further proofs.

A loose tree-decomposition $\mathcal{D} = (T, \chi)$ of width $k \in \mathbb{N}$ is *full* if for every node $t \in V(T)$, $|\chi(t)| = k$ and if for every pair of adjacent nodes t and t' in T, $\chi(t) \ominus \chi(t') = \{x, x'\}$ with $x \in \chi(t)$ and $x' \in \chi(t')$. The next lemma shows that every loose tree-decomposition can be turned into a full one of the same width.

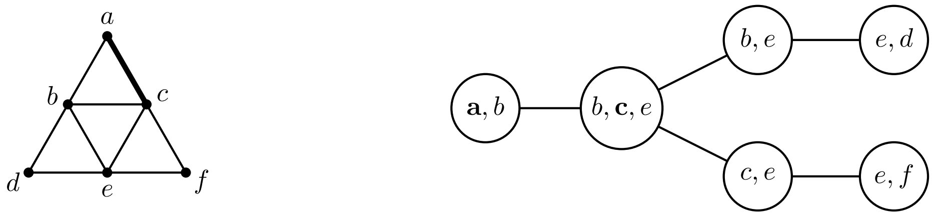

Figure 2: A loose tree-decomposition (on the right) $\mathcal{D}$ of the 3-sun G (graph on the left). Observe that the edge ac is a marginal edge as no bag contains the two vertices a and c, they are however contained in two adjacent bags. Note that bd and cf are also marginal edges. We have $\mathsf{width}(\mathcal{D}, G) = 3$. Notice that if $G' = G - \{a\}$, then G' has a loose tree decomposition of width 2. This decomposition is obtained from $\mathcal{D}$ by removing the bags $\{a, b\}$ and $\{b, c, e\}$ and by making adjacent the nodes corresponding to the bags $\{b, e\}$ and $\{c, e\}$. Notice that in this new loose tree decomposition, the edge bc is a marginal edge.

Lemma 1. *Given a loose tree-decomposition $\mathcal{D} = (T, \chi)$ of a graph G, one can compute a full loose tree-decomposition $\mathcal{D}' = (T', \chi')$ of G such that $\mathsf{width}(\mathcal{D}, G) = \mathsf{width}(\mathcal{D}', G)$.*

Proof. Suppose that $\mathsf{width}(\mathcal{D}, G) = k$. We can assume that in $\mathcal{D}$, if two nodes t, t' of T are adjacent, then neither $\chi(t) \subseteq \chi(t')$ nor $\chi(t') \subseteq \chi(t)$. Let $t \in V(T)$ be a node such that $|\chi(t)| = k$. Suppose there exists a node $t' \in V(T)$ adjacent to t such that $|\chi(t')| < k$. As there is at most one marginal edge e between t and t', it is possible to add vertices from $\chi(t)$ that are not incident to e to complete $\chi(t')$ to a bag of size k. Observe that completing every bag of T that way does not violate the conditions of Definition 2. We can process all the nodes of $\mathcal{D}$ to obtain a loose tree-decomposition whose bags all have size k. Let us assume it is the case for $\mathcal{D}$.

Suppose that t_1 and t_2 are adjacent nodes in T such that $|\chi(t_1) \cap \chi(t_2)| < k - 1$. So there exists $x_1, y_1 \in \chi(t_1) \setminus \chi(t_2)$ and some $x_2, y_2 \in \chi(t_2) \setminus \chi(t_1)$. Since there exists at most one marginal edge between t_1 and t_2, we can suppose without loss of generality that $x_1x_2 \notin E(G)$. Then we can remove from T the edge t_1t_2 and insert a new node t adjacent to t_1 and t_2 such that $\chi(t) = (\chi(t_1) \cup \chi(t_2)) \setminus \{x_1, x_2\}$. Again, one can observe that this transformation does not violate the conditions of Definition 2. It follows that processing T as long as it contains a tree-edge on which this transformation applies, yields to transform $\mathcal{D}$ into a full loose tree-decomposition $\mathcal{D}$ without increasing the width. □

Lemma 2. *Let $\mathcal{D} = (T, \chi)$ be a full loose tree-decomposition of a graph G. For every node t that is not a leaf of T, $\chi(t)$ is a separator of G.*

Proof. As t is not a leaf, T contains two nodes t_1 and t_2 adjacent to t. As $\mathcal{D}$ is full, there exist $x_1 \in \chi(t_1) \setminus \chi(t)$ and $x_2 \in \chi(t_2) \setminus \chi(t)$. Let P be a path in G between x_1 and x_2. Let us recall that every edge of G is either a marginal edge between two neighboring nodes of T or incident to two vertices belonging to the bag of some node of T. This implies that the set of nodes $\{t \in V(T) \mid t \in T_x, x \in P\}$ induces a connected subtree of T. As t_1 and t_2 belong to that set, so does t. It follows that P intersects $\chi(t)$ and thereby $\chi(t)$ is a separator of G. □

Lemma 3. *Let $\mathcal{D} = (T, \chi)$ be a full loose tree-decomposition of a graph G and let $\{t_1, t_2\}$ be an edge of T. If $\chi(t_1) \ominus \chi(t_2) = \{x_1, x_2\} \notin E(G)$, then $\chi(t_1) \cap \chi(t_2)$ is a separator of G.*

Proof. Suppose that $x_1 \in \chi(t_1) \setminus \chi(t_2)$ and $x_2 \in \chi(t_2) \setminus \chi(t_1)$. We prove that $\chi(t_1) \cap \chi(t_2)$ is a (x_1, x_2)-separator. Let P be a path between x_1 and x_2 in G. We let T_1 denote the largest subtree of T containing t_1 but not t_2. Consider the vertex set $V_1 = \{x \in V(G) \mid \exists t \in T_1, x \in \chi(t)\}$. Let $x \in V_1$ and $y \notin V_1$ be the two adjacent vertices of P such that every vertex z of P between x_1 and x belongs to V_1. As $x_2 \notin V_1$, the vertices x and y are well defined (x may be equal to x_1). By Definition 2, every edge is either a marginal edge between two adjacent nodes of T or is contained in the bag of some node of T. By assumption, there is no marginal edge between t_1 and t_2. As $x \in V_1$ and $y \notin V_1$, we have that $x \in \chi(t_1) \cap \chi(t_2)$, and thereby $\chi(t_1) \cap \chi(t_2)$ intersects P. □

2.4 Tight brambles

In order to define a notion of obstacle to the existence of a loose tree-decomposition of small width, we now adapt the well-known definition of bramble that is used in the context of tree-decomposition and treewidth [19]. Let G be a graph. Two subsets S_1 and S_2 of $V(G)$ are *tightly touching* if either $S_1 \cap S_2 \neq \emptyset$ or $E(G)$ contains two distinct edges x_1x_2 and y_1y_2 such that $x_1, y_1 \in S_1$ and $x_2, y_2 \in S_2$ (the edges x_1x_2 and y_1y_2 may share one vertex but not two, as graphs are considered to be simple). For simplicity, in this paper we use *touching* as a shortcut of the term *tightly touching.*

Definition 3 (Tight bramble)**.** *Let G be a graph. A set $\mathcal{B} \subseteq 2^{V(G)}$ of pairwise touching connected subsets of vertices, each of size at least two, is a* tight bramble *of G. A set $S \subseteq V(G)$ is a* cover *of $\mathcal{B}$ if for every set $B \in \mathcal{B}$, $S \cap B \neq \emptyset$. The* order *of the bramble $\mathcal{B}$ is the smallest size of a cover of $\mathcal{B}$.*

Figure 3: On the left, we observe that $\mathcal{B} = \big\{\{a,b\},\{b,c\},\{c,d,e,f,a\}\big\}$ is a tight bramble of order two. On the right, $\mathcal{B}_1 = \big\{\{x,y\} \mid xy \in E(G)\big\}$ is a tight bramble of order 4 of the K_4. Observe that $\mathcal{B}_2 = \big\{\{a\},\{b,c\},\{c,d\},\{b,c\}\big\}$ is a set of pairwise touching connected subsets of $V(K_4)$. However, according to Definition 3, as it contains the singleton set $\{a\}$, is it not a tight bramble. See Observation 2 and the related discussion.

Let us discuss Definition 3. Observe first that, if G is an edge-less graph, then its unique tight bramble is $\mathcal{B} = \emptyset$ which has order 0. In the rest of the paper, we will not consider edge-less graphs as searching a fugitive (located on edges) in such a graph would create degenerate situations. The next observation provides a characterization of trees by means of tight brambles.

Observation 1. *Let G be a connected graph. The maximum order of a bramble of G is 1 if and only if G is a tree.*

Proof. We observe that if G is a tree, a tight bramble consists of set of pairwise intersecting subtrees of G. It follows by he Helly property that the maximum order of a bramble of a tree is 1. Suppose that G is not a tree. Then G contains a cycle C of at least 3 edges. Let $f_1, f_2, \dots f_t$, $t \geq 3$, be the edges the edges of C such that for $1 \leq i < t$, $f_i \cap f_{i+1} \neq \emptyset$ and $f_1 \cap f_t \neq \emptyset$. Then observe that $\mathcal{B} = \big\{\{f_1\},\{f_2\}, C \setminus \{f_1, f_2\}\big\}$ is a tight bramble of order 2 (see Figure 3). ☐

Finally, if we discard edge-less graphs, then we can observe that the size-two constraint on the elements of a tight bramble can be relaxed without changing the value of the maximum order of a tight bramble of a given graph.

Observation 2. *Let $\mathcal{B} \subset 2^{V(G)}$ be a set of pairwise touching connected subsets of $V(G)$ containing the singleton set $\{x\}$, for some $x \in V(G)$. If the order of $\mathcal{B}$ is k, then there exists a tight bramble $\mathcal{B}'$ of G of order k.*

Proof. To construct $\mathcal{B}'$, we proceed as follows. For every $S \in \mathcal{B}$ such that $S \neq \{x\}$, let y_S and z_S be two vertices of S such that $xy_S \in E$ and $xz_S \in S$. Then we set $\mathcal{B}' = \big(\mathcal{B} \setminus \{x\}\big) \bigcup \big\{\{xy_S\},\{xz_S\} \mid S \in \mathcal{B}, S \neq \{x\}\big\}$. In other words, $\mathcal{B}'$ is obtained by replacing $\{x\}$ by every pair of edges $\{xy_S\}$ and $\{xz_S\}$. Clearly, the order of $\mathcal{B}'$ is at most k. To see this, consider a cover X of $\mathcal{B}$. Observe that $x \in X$ and thereby X is a cover of $\mathcal{B}'$ as well. Now assume $\mathcal{B}'$ has a cover X' not containing X. Let us consider $S \in \mathcal{B} \cap \mathcal{B}'$. Then S contains two vertices of S, namely y_S and z_S. It is easy to see that $X' \setminus \{y_S\} \cup \{x\}$ is also a cover of $\mathcal{B}'$. It follows that $\mathcal{B}'$ has a cover of minimum order that contains x and that is thereby also a cover of $\mathcal{B}$. ☐

Lemma 4. *Let $\mathcal{B}$ be a tight bramble of a graph G and let S_1 and S_2 be two covers of $\mathcal{B}$. If $S \subsetneq V(G)$ separates S_1 from S_2 in G, then S is a cover of $\mathcal{B}$.*

Proof. Let $\mathcal{B}$ be a tight bramble of a graph G and S_1, S_2 two covers of $\mathcal{B}$. Let $S \subsetneq V(G)$ be a separator of S_1 and S_2. Consider a set $B \in \mathcal{B}$ of the tight bramble. Since S_1 and S_2 cover B and since B is connected, there exists a path P whose internal vertices belong to B and whose endpoints belong to S_1 and S_2. Thus, since S separates S_1 and S_2, there exists a vertex $x \in P$ such that $x \in S$ and thus, S covers B. This holds for every $B \in \mathcal{B}$, thus S is a cover of $\mathcal{B}$. □

2.5 Mixed search games

In this paper, we deal with mixed search games on a graph G introduced by Bienstock and Seymour [4]. The opponents are a team of searchers and a fugitive. We assume that both players have full knowledge of the graph and the position of their opponent in the graph. Moreover, the fugitive is *agile* in the sense that it may move at any moment though unguarded pathways. At each round, the searchers occupy a subset of vertices, called *searchers' position*, while the fugitive is located on an edge, called *fugitive location*. A *play* is a (finite or infinite) sequence alternating between searchers' positions and fugitive locations, that is a sequence $\mathcal{P} = \langle S_0, e_1, S_1, \dots, e_\ell, S_\ell, \dots \rangle$ with $S_0 = \emptyset$ and where for $i \geq 1$, $S_i \subseteq V(G)$ is a searchers' position and $e_i \in E(G)$ is a fugitive location. In case a play is finite, it always ends with the special symbol $\star$, replacing a fugitive location and indicating the fact that the fugitive has been captured. The *cost* of a play $\mathcal{P}$ is the maximum size of a searchers' position S_i in it and is denoted by $\mathsf{cost}(\mathcal{P})$.

The search strategy. A *searchers' move* is a pair $(S, S') \in \left(2^{V(G)}\right)^2$ indicating a transition from the searchers' position S to searcher's position S'. In a mixed search game, a searchers' move is *legitimate* if $|S \ominus S'| \in \{1, 2\}$ and, moreover, if $|S \ominus S'| = 2$ then $S \ominus S'$ is an edge of G. This allows three types of moves:

- [*Placement of a searcher*]: $S' = S \cup \{x\}$ for some $x \in V(G) \setminus S$. This type of move consists in placing a new searcher on vertex x.
- [*Removal of a searcher*]: $S = S' \setminus \{x\}$ for some $x \in S$. This type of move consists in removing a searcher from vertex x.
- [*Sliding of a searcher along an edge*]: $S \ominus S' = \{x, y\}$ and $xy \in E(G)$. This type of move corresponds to sliding the searcher initially positioned at vertex x, along the edge xy, towards eventually occupying vertex y.

As the searchers know the exact fugitive location, they can use this information to determine their next positions. Therefore, we may define a *(mixed) search strategy* on G as a function $\mathbf{s}_G \in \left(2^{V(G)}\right)^{\left(2^{V(G)} \times E(G)\right)}$ such that, for every $(S, e) \in 2^{V(G)} \times E(G)$, the pair $\big(S, \mathbf{s}_G(S, e)\big)$ is a legitimate searchers' move. We denote by $\mathcal{S}_G$ the set of all search strategies on G. Every legitimate searchers' move (S, S') immediately clears a set of edges defined as follows:

$$\mathsf{clear}_G(S, S') := \begin{cases} \big\{xy \mid x \in S\big\} \cap E(G), & \text{if } S' \setminus S = \{y\} \\ \emptyset & \text{if } S' \setminus S = \emptyset. \end{cases}$$

The fugitive strategy. To formally define the fugitive strategy we need to introduce some concepts. Let us consider a legitimate searchers' move (S, S').

A *pathway* $\mathcal{W} = \langle f_1, f_2, \dots f_t \rangle$ of G, with $t \geq 2$, is (S, S')-*avoiding* if the following conditions are satisfied:

1. for every $j \in [2, t]$, $f_{j-1} \cap f_j \notin S \cap S'$, and
2. if $S' \setminus \{y\} = S \setminus \{x\}$ (that is, (S, S') is a sliding move from x to y along the edge xy, see Figure 4), and $xy \in \mathcal{W}$, then

- $xy = f_1$ or $xy = f_t$;
- if $xy = f_1 = f_t$, then $t > 2$.

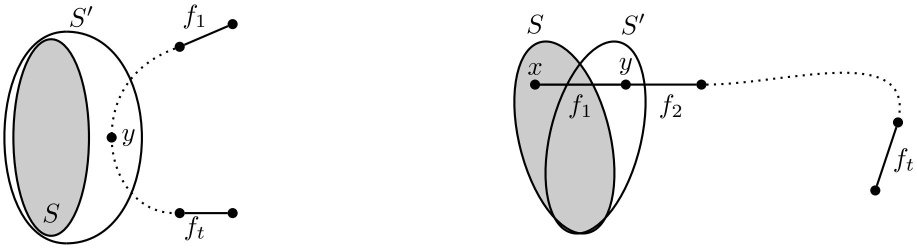

Figure 4: Two legitimate moves (S, S') and (S, S')-avoiding pathway $\mathcal{W} = \langle f_1, f_2, \dots f_t \rangle$ are depicted. On the left, we have $S' = S \cup \{x\}$, and $\mathcal{W}$ may go through the vertex x. On the right, $S \ominus S' = \{x, y\}$, i.e., a searcher slides along the edge $f_1 = xy$ from x to y. Observe that, if $t > 2$, then the edge f_t may be the edge f_1.

Given a legitimate searchers' move (S, S') and an edge e, the subset of edges of $E(G)$ that are *accessible* from e through a (S, S')-avoiding pathway of G is defined as

$$A_G(S, e, S') := \left\{ e' \in E(G) \setminus \binom{S'}{2} \mid \text{there is an } (S, S')\text{-avoiding } (e, e')\text{-pathway} \right\}.$$

We can now define the *fugitive space* so as to contain the edges where the fugitive initially residing at e may move after the searchers' move (S, S'), that is

$$\mathsf{fsp}_G(S, e, S') := (\{e\} \setminus \mathsf{clear}_G(S, S')) \cup A_G(S, e, S').$$

Given a graph G, we set $E^\star(G) = E(G) \cup \{\star\}$ and we define a *fugitive strategy* on G as a pair $(e_1, \mathbf{f}_G) \in E(G) \times \left(E^\star(G)\right)^{\left(2^{V(G)} \times E(G) \times 2^{V(G)}\right)}$ such that, for the function $\mathbf{f}_G$, if $\mathsf{fsp}_G(S, e, S') \neq \emptyset$, then $\mathbf{f}_G(S, e, S') \in \mathsf{fsp}_G(S, e, S')$, otherwise $\mathbf{f}_G(S, e, S') = \star$. We denote by $\mathcal{F}_G$ the set if all fugitive strategies on G.

Winning and monotone search strategies A search program (on a graph G) is a pair $\left(\mathbf{s}_G, (e_1, \mathbf{f}_G)\right) \in \mathcal{S}_G \times \mathcal{F}_G$. Every search program $\left(\mathbf{s}_G, (e_1, \mathbf{f}_G)\right)$ *generates* a play, denoted

$$\mathcal{P}(\mathbf{s}_G, e_1, \mathbf{f}_G) \quad := \quad \langle S_0, e_1, S_1, \dots, e_\ell, S_\ell, \dots \rangle \tag{1}$$

where for each $i \geq 1$, $S_i = \mathbf{s}_G(S_{i-1}, e_i)$ and $e_{i+1} = \mathbf{f}_G(S_{i-1}, e_i, S_i)$. The search program $(\mathbf{s}_G, e_1, \mathbf{f}_G)$ is *monotone* if, in (1), for every $i \geq 1$, the edge e_i has not been cleared at any step prior to i, that is for every $j \leq i$, $e_i \notin \mathsf{clear}_G(S_{j-1}, S_j)$. A search strategy $\mathbf{s}_G \in \mathcal{S}_G$ is *monotone* if for every fugitive strategy $(e_1, \mathbf{f}_G) \in \mathcal{F}_G$, the program $\left(\mathbf{s}_G, (e_1, \mathbf{f}_G)\right)$ is monotone.

Let $\mathbf{s}_G$ be a search strategy. The *cost*, denoted by $\mathsf{cost}(\mathbf{s}_G)$, of $\mathbf{s}_G$ is the maximum cost of $\mathcal{P}(\mathbf{s}_G, e_1, \mathbf{f}_G)$, over all $(e_1, \mathbf{f}_G) \in \mathcal{F}_G$. Also, $\mathbf{s}_G$ is *winning* if for every fugitive strategy $(e_1, \mathbf{f}_G) \in \mathcal{F}_G$, the play $\mathcal{P}(\mathbf{s}_G, e_1, \mathbf{f}_G)$ is finite.

We define the *mixed search number (against an agile and visible fugitive)* of G as:

$$\mathsf{avms}(G) = \min \left\{ \mathsf{cost}(\mathbf{s}_G) \mid \mathbf{s}_G \text{ is a winning search strategy in } \mathcal{S}_G \right\}.$$

And if we restrict the search strategies to the monotone ones, we define:

$$\mathsf{mavms}(G) = \min \left\{ \mathsf{cost}(\mathbf{s}_G) \mid \mathbf{s}_G \text{ is a monotone winning search strategy in } \mathcal{S}_G \right\}.$$

3 Monotonicity of the mixed search game against an agile and visible fugitive

This section is devoted to the main result of this paper, that is Theorem 1 below. It provides a min-max characterization of graphs with Cartesian tree product number at most k in terms of tight bramble obstacles. Moreover, it shows that the Cartesian tree product number corresponds to the number of searchers required to capture an agile and visible fugitive in a mixed search game. As a byproduct, it proves that the mixed search game against an agile and visible fugitive is monotone. This results can be seen as the "visible counterpart" of the result of Bienstock and Seymour [4] where the monotonicity is proved for the case of an agile and invisible fugitive. Also, it can be seen as the "mixed counterpart" of the result of Seymour and Thomas [19] where the monotonicity is proved for the *node* search game against an agile and visible fugitive.

Theorem 1. *Let G be a graph and $k \geq 1$ be an integer. Then the following conditions are equivalent:*

1. *G has a loose tree-decomposition of width k;*
2. *G is a minor of $T^{(k)} = T \square K_k$ (*i.e. $\mathsf{ctp}(G) \leq k$*);*
3. *every tight bramble $\mathcal{B}$ of G has order at most k;*
4. *the mixed search number against an agile and visible fugitive is at most k (*i.e. $\mathsf{avms}(G) \leq k$*);*
5. *the monotone mixed search number against an agile and visible fugitive is at most k (*i.e. $\mathsf{mavms}(G) \leq k$*).*

The proof of Theorem 1 is as follows. First, Lemma 6 proves that ($1 \Leftrightarrow 2$). Then Lemma 8 establishes ($3 \Rightarrow 2$). In turn Lemma 10 shows ($2 \Rightarrow 5$). The fact that ($5 \Rightarrow 4$) is trivial. Finally, we can conclude, using Lemma 9, that ($4 \Rightarrow 3$).

3.1 The Cartesian tree product number and loose-tree decomposition

Lemma 6 establishes that the minimum width of a loose tree-decomposition is equal to the *Cartesian tree product number* of a graph (($1 \Leftrightarrow 2$) from Theorem 1). We first show that every minor of $T^{(k)} = T \square K_k$ admits a loose tree-decomposition of width k. To that aim, we prove that the width of a loose tree-decomposition of a graph is a parameter that is closed under the minor relation (Lemma 5). Then, we build for $T^{(k)}$ a loose tree-decomposition of width at most k. To prove the reverse direction, we show that given a loose tree-decomposition $\mathcal{D}$ of width k of G, one can complete the graph G into a graph H which is contained in $T^{(k)}$ as a minor. The graph H is obtained by adding all possible missing edge between pair of vertices of the same bag of $\mathcal{D}$ and all possible missing marginal edges of $\mathcal{D}$.

Lemma 5. *Let G and H be two graphs such that $H \preceq G$. If G has a loose tree-decomposition of width k, then also does H.*

Proof. Let $\mathcal{D} = (T, \chi)$ be a loose tree-decomposition of G. Suppose that H is obtained by removing an edge or a vertex from a graph G. It is clear from Definition 2 that restricting the bags of $\mathcal{D}$ to the vertices of H yields a loose tree-decomposition of H and does not increase the width. So suppose that $H = G_{/e}$ from some edge $e = xy \in E(G)$. Let v_e be the new vertex resulting from the contraction of e. We define $\mathcal{D}' = (T, \chi')$ such that for every $t \in V(T)$, if $x, y \notin \chi(t)$, then $\chi'(t) = \chi(t)$, otherwise $\chi'(t) = \chi(t) \setminus \{x, y\} \cup \{v_e\}$. We prove that $\mathcal{D}'$ is a loose tree decomposition of H.

- First observe that by condition ($\mathsf{L2}$), in $\mathcal{D}$ the trace T_x of x and the trace T_y of y either intersect or are joined by a tree edge. It follows that in $\mathcal{D}'$, the trace T_{v_e} is connected. So condition ($\mathsf{L1}$) holds for $\mathcal{D}'$.
- By construction of $\mathcal{D}'$, for every node $t \in V(T)$ such that $x \in \chi(t)$ or $y \in \chi(t)$, we have $v_e \in \chi'(t)$. This implies that $\mathcal{D}'$ satisfies condition ($\mathsf{L2}$).

- Suppose that in $\mathcal{D}'$, the edge $uv \in E(H)$ is a marginal edge for the nodes t_1 and t_2 of T. This means that $u \in \chi'(t_1) \setminus \chi'(t_2)$ and $v \in \chi'(t_2) \setminus \chi'(t_1)$. By construction, if u and v both belong to $V(G)$, then uv is also a marginal edge of G for the nodes t_1 and t_2 in $\mathcal{D}$. So assume, without loss of generality that say $u = v_e$. As v_e results from the contraction of the edge $xy \in E(G)$, G contains the edge xv or yv. Suppose that $xv \in E$. By construction of $\mathcal{D}'$, we have $x \in \chi(t_1) \setminus \chi(t_2)$. It follows that in $\mathcal{D}$, xv is a marginal edge between the nodes t_1 and t_2 of T. So we proved that every marginal edge in $\mathcal{D}'$ corresponds to a marginal edge in $\mathcal{D}$. This implies that if for some tree-edge t_1t_2 of T, $\mathcal{D}'$ does not satisfies condition (L3), neither does $\mathcal{D}$.

Finally, we observe that, by construction, $\mathsf{width}(\mathcal{D}, H) \leq \mathsf{width}(\mathcal{D}, G)$, concluding the proof. $\square$

Lemma 6. *For every graph G, we have*

$$\mathsf{ctp}(G) = \min\{\mathsf{width}(\mathcal{D}, G) \mid \mathcal{D} \text{ is a loose tree-decomposition of } G\}.$$

Proof. We first prove that $\min\{\mathsf{width}(\mathcal{D}, G) \mid \mathcal{D} \text{ is a loose tree-decomposition of } G\} \leq \mathsf{ctp}(G)$. By Lemma 5, it is enough to prove the statement for $T^{(k)}$, where $k = \mathsf{ctp}(G)$. Recall that, by construction of $T^{(k)}$, we have that $V(T^{(k)}) = \{x_{(t,i)} \mid t \in V(T), i \in [k]\}$ and $E(T^{(k)}) = \{x_{(t,i)}x_{(t,j)} \mid t \in V(T), i \neq j\} \cup \{x_{(t,i)}x_{(t',i)} \mid t, t' \in V(T), t \neq t', i \in [k]\}$. We build a loose tree-decomposition $\mathcal{D} = (T', \chi)$ of $T^{(k)}$ as follows:

- For the sake of the construction of T', we fix an arbitrary node r of T as the root of T, implying that for every node $t \in V(T)$ distinct from r, there is a uniquely defined parent node $p(t)$. The tree T' is obtained from T by every edge $k-1$ times. More formally, we define (see Figure 5):

$$V(T') = \{v_{(r,k)}\} \cup \{v_{(t,j)} \mid t \in V(T), t \neq r, j \in [k]\}, \text{ and}$$

$$E(T') = \{v_{(t,j)}v_{(t,j+1)} \mid t \in V(T), t \neq r, j \in [k-1]\} \cup \{v_{(p(t),k)}v_{(t,1)} \mid t \in V(T), t \neq r\}.$$

- We set $\chi(v_{(r,k)}) = \{x_{(r,i)} \mid i \in [k]\}$ and for every node $v_{(t,j)} \in V(T')$ such that $t \neq r$, we set

$$\chi(v_{(t,j)}) = \{x_{(t,j')} \mid 1 \leq j' \leq j\} \cup \{x_{(p(t),j')} \mid j < j' \leq k\}.$$

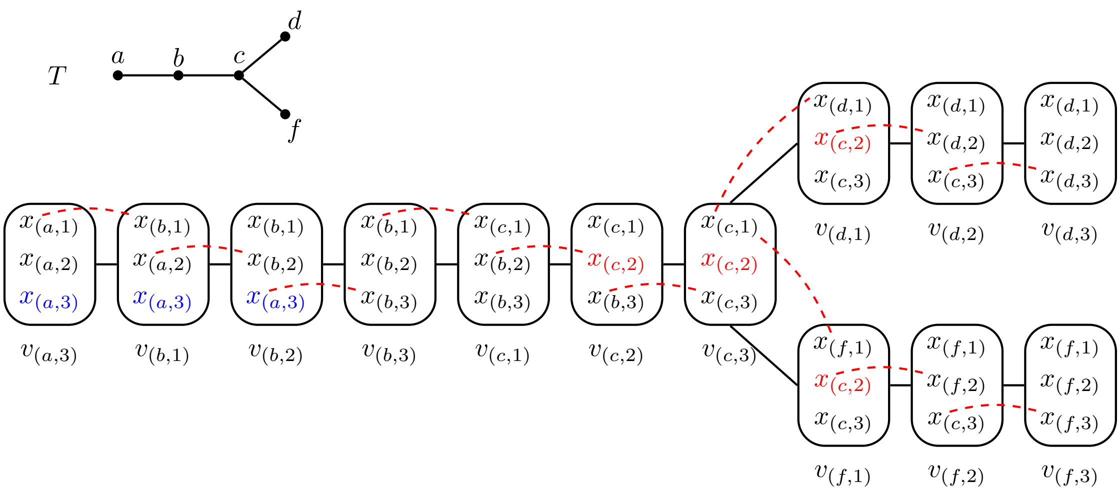

Figure 5: A loose tree-decomposition $\mathcal{D} = (T', \chi)$ of $T^{(3)} = T \square K_3$. To build $\mathcal{D}$, we set node a as the root of T. In blue, we have the trace $T'_{x_{(a,3)}}$ and in red the trace $T'_{x_{(c,2)}}$. The dashed red edges are the marginal edges of $\mathcal{D}$.

One can observe that by construction of $\mathcal{D}$, the trace $T_{x_{(t,i)}}$ of every vertex is connected. Therefore condition (L1) of Definition 2 holds. We observe that $T^{(k)}$ contains two types of edges. For an edge

$e \in E(T^{(k)})$, either $e = x_{(t,i)}x_{(t,i')}$ for some node $t \in V(T)$ and some distinct integers $i, i' \in [k]$, or $e = x_{(t,i)}x_{(p(t),i)}$ where $t \in V(T)$ is not the root node and $i \in [k]$. In the former case, $x_{(t,i)}$ and $x_{(t,i')}$ both belong to the bag $\chi(v_{(t,k)})$. In the later case, we observe that: 1) if $i = 1$, then $x_{(t,i)} \in \chi(v_{(t,1)})$ and $x_{(p(t),i)} \in \chi(v_{(p(t),k)})$; 2) otherwise $x_{(t,i)} \in \chi(v_{(t,i)})$ and $x_{(p(t),i)} \in \chi(v_{(t,i-1)})$. It follows that condition (L2) of Definition 2 holds. Finally, observe that only the edges of $T^{(k)}$ of the second type are not covered by a single bag. These are the marginal edges and by construction, there is at most one marginal edge between two adjacent nodes in $\mathcal{D}$. It follows that condition (L3) of Definition 2 also holds and thereby $\mathcal{D}$ is a loose tree-decomposition of $T^{(k)}$. Finally, notice that $\mathsf{width}(\mathcal{D}, T^{(k)}) = k$, proving that $\min\{\mathsf{width}(\mathcal{D}, T^{(k)}) \mid \mathcal{D} \text{ is a loose tree-decomposition of } T^{(k)}\} \leq \mathsf{ctp}(T^{(k)})$.

Let us now prove that $\mathsf{ctp}(G) \leq \min\{\mathsf{width}(\mathcal{D}, G) \mid \mathcal{D} \text{ is a loose tree-decomposition of } G\}$. By Lemma 1, we can assume that $\mathcal{D} = (T, \chi)$ is a full loose tree-decomposition of G such that $\mathsf{width}(\mathcal{D}, G) = k$. We prove that $G \preceq T^{(k)}$. Based on $\mathcal{D}$, we construct a graph H by adding edges to G as follows:

$$\begin{aligned} E(H) = \quad & \{xy \mid \exists t \in V(T), \{x, y\} \subseteq \chi(t)\} \bigcup \\ & \{xy \mid \exists t_1t_2 \in E(T), x \in \chi(t_1) \setminus \chi(t_2), y \in \chi(t_2) \setminus \chi(t_1)\} \end{aligned}$$

That is the edge set of H contains all possible edges between vertices belonging to a common bag of $\mathcal{D}$ and all possible marginal edges between adjacent bags. Clearly $E(H)$ contains $E(G)$. So, let us prove that $H \preceq T^{(k)}$. To that aim, we iteratively build a minor-model ρ of H in $T^{(k)}$ with the following algorithm.

Initially, we set $\rho(x) = \emptyset$ for every vertex $x \in V(H)$. We guarantee that, at each step of the algorithm, if $x_{(t,i)} \in \rho(x)$ and $x_{(t,j)} \in \rho(x)$, then $i = j$. We pick an arbitrary node $t \in V(T)$. For every $x \in \chi(t)$, we set $\rho(x) = \{x_{(t,i)}\}$ (with $i \in [k]$) in a way that, for $y \in \chi(t)$, if $x \neq y$, $\rho(x) \neq \rho(y)$. Set $S = \{t\}$ as the set of processed nodes of T (a node t is processed when for every vertex $x \in \chi(t)$, $\rho(x)$ is defined). Assume there exists an unprocessed node $t \notin S$. We pick a node $t' \notin S$ having an adjacent node $t \in S$. Suppose that $x \in \chi(t) \cap \chi(t')$. Then $\rho(x)$ contains a vertex $x_{(t,i)}$ for some $i \in [k]$. We add $x_{(t',i)}$ to $\rho(x)$. Otherwise, suppose that $x \in \chi(t') \setminus \chi(t)$. As $|\chi(t) \cap \chi(t')| = k-1$, there is a unique vertex $y \in \chi(t) \setminus \chi(t')$. Let $j \in [k]$ be such that $x_{(t,j)} \in \rho(y)$. Then we add $x_{(t',j)}$ to $\rho(x)$.

First observe that by construction, for every $x \in V(H)$, the set $\rho(x)$ is a connected subset of $T^{(k)}$. Let us argue that for every $xy \in E(H)$, $T^{(k)}$ contains an edge between a vertex of $\rho(x)$ and a vertex of $\rho(y)$. As explained in the description of H, $E(H)$ contains two type of edges xy: either $V(T)$ contains a node t such that $xy \subseteq \chi(t)$ or xy is a marginal edge, that is $E(T)$ contains a tree-edge t_1, t_2 such that $x \in \chi(t_1) \setminus \chi(t_2)$ and $y \in \chi(t_2) \setminus \chi(t_1)$. In the former case, by the construction of $\rho(x)$, there exist distinct $i, i' \in [k]$ such that $x_{(t,i)} \in \rho(x)$ and $x_{(t,i')} \in \rho(y)$. Observe that by definition of $T^{(k)}$, $x_{(t,i)}x_{(t,i')} \in E(T^{(k)})$. In the latter case, there exist in T two adjacent nodes t_1 and t_2 such that $x \in \chi(t_1) \setminus \chi(t_2)$, $y \in \chi(t_2) \setminus \chi(t_1)$. Again, by the construction of $\rho(x)$ and $\rho(y)$, there exists a unique $i \in [k]$ such that $x_{(t,i)} \in \rho(x)$ and $x_{(t',i)} \in \rho(y)$. Observe that $x_{(t,i)}x_{(t',i)} \in E(T^{(k)})$. It follows that ρ certifies that H is a minor of $T^{(k)}$. As G is a subgraph of H, G is a minor of $T^{(k)}$ and thereby $\mathsf{ctp}(G) \leq k$. $\square$

3.2 Tight brambles certify large Cartesian tree product number

In this subsection, we show (Lemma 8) that the existence of tight bramble of order k in a graph G certifies the fact that the Cartesian tree product number of G is at most k ((3 $\Rightarrow$ 2) of Theorem 1). Our proof follows the lines of the proof of Bellenbaum and Diestel [2] where it is shown that, in the context of treewidth, a bramble of large order is an obstacle to a small treewidth. We first prove a technical lemma showing that extending in a connected way the trace T_x of a vertex x in a loose tree-decomposition $\mathcal{D} = (T, \chi)$ of a graph G yields another loose tree-decomposition of G.

Lemma 7. *Let $\mathcal{D} = (T, \chi)$ be a loose tree-decomposition of a graph G. Let $\chi' : V(T) \to 2^{V(G)}$ be such that for every $x \in V(G)$, $\{t \in V(T) \mid x \in \chi'(t)\}$ induces a connected subtree T'_x of T and the trace T_x is a subtree of T'_x, then $\mathcal{D}' = (T, \chi')$ is a loose tree-decomposition of G.*

Proof. Let us show that the conditions (L1), (L2) and (L3) of Definition 2 are satisfied by $\mathcal{D}'$. Clearly, (L1) is verified by definition of T'_x. Suppose xy is an edge in $E(G)$. As $\mathcal{D}$ is a loose tree-decomposition, there exists a tree-edge $f = t_1t_2 \in E(T)$ such that $xy \in E\big(G[\chi(t_1) \cup \chi(t_2)]\big)$. As for every vertex x, T_x is a subtree of T'_x, we have $xy \in E\big(G[\chi'(t_1) \cup \chi'(t_2)]\big)$. It follows that (L2) holds. Finally, observe that as, by constuction of $\mathcal{D}'$, for every vertex x, the trace T_x is a subtree of T'_x, every marginal edge in $\mathcal{D}'$ is a marginal edge in $\mathcal{D}$, implying that $\mathcal{D}'$ satisfies (L3). $\square$

Given a tight bramble $\mathcal{B}$ of a graph G of order at most k, a loose tree-decomposition $\mathcal{D} = (T, \chi)$ of G is $\mathcal{B}$-*admissible* if for every node $t \in V(T)$ such that $|\chi(t)| > k$, the bag $\chi(t)$ fails to cover $\mathcal{B}$.

Lemma 8. *Let G be a graph and $k \geq 1$ be an integer. If the maximum order of a tight bramble of G is k, then $\mathsf{ctp}(G) \leq k$.*

Proof. We prove that if G does not contain a tight bramble of order $k+1$ or larger, then for every tight bramble $\mathcal{B}$ of G, there exists a $\mathcal{B}$-admissible loose tree-decomposition of G. This statement implies $\mathsf{ctp}(G) \leq k$. Indeed, $\mathcal{B} = \emptyset$ is a tight bramble. As every subset of vertices covers the empty bramble, a $\emptyset$-admissible loose tree-decomposition cannot contain a bag of size larger than k. It follows by Lemma 6 that $\mathsf{ctp}(G) \leq k$.

Let $\mathcal{B}$ be a non-empty tight bramble of G and let X be a cover of $\mathcal{B}$ of minimum size. We proceed by induction on the size of $\mathcal{B}$.

For the induction base, suppose that the size of $\mathcal{B}$ is maximum. Let $\mathsf{cc}(G-X)$ be the set of connected components of $G-X$. We build a $\mathcal{B}$-admissible loose tree-decomposition $\mathcal{D} = (T, \chi)$ of G as follows. The tree T is a star with $|\mathsf{cc}(G-X)|$ leaves. More precisely, we have $V(T) = \{t_X\} \cup \big\{t_C \mid C \in \mathsf{cc}(G-X)\big\}$ and $E(T) = \big\{t_Xt_C \mid C \in \mathsf{cc}(X)\big\}$. We set $\chi(t_X) = X$. It remains to define the bag $\chi(t_C)$ for every connected component $C \in \mathsf{cc}(G-X)$. Observe that since $C \cap X = \emptyset$ and since X is a cover of $\mathcal{B}$, $C \notin \mathcal{B}$. Moreover, as $\mathcal{B}$ has maximum size, $\mathcal{B} \cup \big\{\{C\}\big\}$ is not a tight bramble of G. This implies that $\mathcal{B}$ contains a set B such that B and C do not touch. It follows that $B \cap C = \emptyset$ and G contains at most one edge uv such that $u \in B \setminus C$ and $v \in C \setminus B$. If such an edge uv exists, then we set $\chi(t_C) = (C \cup N_G(C)) \setminus \{u\}$. Otherwise, we set $\chi(t_C) = C \cup N_G(C)$. In both cases, we have $\chi(t_C) \cap B = \emptyset$ and henceforth $\chi(t_C)$ does not cover $\mathcal{B}$. So if $\mathcal{D} = (T, \chi)$ is a loose tree-decomposition of G, then it is a $\mathcal{B}$-admissible one. Let us prove that the three conditions of Definition 2 are satisfied. Condition (L1) follows from the fact that if a vertex x of G belongs to several bags, then it belongs to $\chi(t_X)$. Condition (L2) holds as $\chi(t_X)$ is a separator and for every connected component $C \in \mathsf{cc}(G-X)$, $\chi(t_C) \subseteq X \cup C$. Finally, condition (L3) is also satisfied because, by construction, if there exists a marginal edge between node t_X and node t_C, then it is unique (as otherwise the sets C and B would touch).

We now consider the case where the size of $\mathcal{B}$ is not maximum. By induction hypothesis, suppose that the statement holds for every tight bramble $\mathcal{B}'$ containing more elements than $\mathcal{B}$. We also assume that for every such larger tight bramble $\mathcal{B}'$, none of the $\mathcal{B}'$-admissible loose tree-decompositions is also $\mathcal{B}$-admissible, as otherwise we are done. Observe that we can assume that $V(G)$ is not a minimum cover of $\mathcal{B}$, as otherwise $|V(G)| \leq k$ and thereby the loose tree-decomposition containing a unique bag $V(G)$ would be $\mathcal{B}$-admissible. By Lemma 6, we then have $\mathsf{ctp}(G) \leq k$. So we assume that $X \neq V(G)$. For a set $S \subseteq V(G)$, we say that a loose tree-decomposition $\mathcal{D} = (T, \chi)$ of G is S-*rooted* if T contains a node t such that $S \subseteq \chi(t)$.

Claim 1. *For every connected component C of $G-X$, there exists a X-rooted loose tree-decomposition $\mathcal{D}_C = (T_C, \chi_C)$ of $H = G[C \cup X]$ such that for every node $t \in V(T_C)$ such that $|\chi_C(t)| \geq k+1$, $\chi(t)$ does not cover $\mathcal{B}$.*

Suppose Claim 1 holds. Since for every $C \in cc(G-X)$, $\mathcal{D}_C = (T_C, \chi_C)$ contains a node t_C such that $X \subseteq \chi_C(t_C)$, these loose tree-decompositions can be amalgamated into a loose tree-decomposition $\mathcal{D} = (T, \chi)$ of G as follows: T contains a node t_X such that $\chi(t_X) = X$ and for each $C \in cc(G-X)$, t_X is made adjacent to the node t_C of T_C.

Proof of Claim 1: Let C be a connected component of $G-X$. We denote $H = G[X \cup C]$ and consider $\mathcal{B}' = \mathcal{B} \cup \{C\}$. We consider two cases:

- The set $\mathcal{B}'$ is not a tight bramble of G. Then $\mathcal{B}$ contains a set B such that B and C are not touching, that is $B \cap C = \emptyset$ and $E(G)$ contains at most one edge e that is incident to a vertex of B and to a vertex of C. We proceed as in the induction base case above. Consider $\mathcal{D}_C = (T_C, \chi_C)$ where T_C is the tree defined on two nodes t_1 and t_2. We set $\chi_C(t_1) = X$. If there is an edge uv such that $u \in B \setminus C$ and $v \in C \setminus B$, then $\chi_c(t_2) = (C \cup N_G(C)) \setminus \{u\}$, otherwise $\chi_c(t_2) = C \cup N_G(C)$. First observe that $\mathcal{D}_C$ is a loose tree-decomposition of H satisfying the statement of the claim. Indeed, by construction, as B and C are not touching. It follows that on the one hand, there is at most one marginal edge between $\chi(t_1)$ and $\chi(t_2)$, and so $\mathcal{D}_C = (T_C, \chi_C)$ is a loose tree-decomposition of H. On the other hand, this implies that $B \cap C = \emptyset$ and so $\chi(t_C)$ does not cover $\mathcal{B}$.

- The set $\mathcal{B}'$ is a tight bramble of G. Since X covers $\mathcal{B}$ and $C \cap X = \emptyset$, $C \notin \mathcal{B}$. Therefore, we have $|\mathcal{B}'| > |\mathcal{B}|$. Consequently, by the induction hypothesis, we can assume the existence of a $\mathcal{B}'$-admissible loose tree-decomposition $\mathcal{D} = (T, \chi)$ of G. By the induction assumption, $\mathcal{D}$ is not $\mathcal{B}$-admissible. This implies that there exists a node $s \in V(T)$ such that $|\chi(s)| \geq k+1$ and $\chi(s)$ covers $\mathcal{B}$. We build from $\mathcal{D}$ a X-rooted loose tree-decomposition $\mathcal{D}_C = (T_C, \chi_C)$ of $H = G[X \cup C]$ as follows (see Figure 6): for every vertex $x \in X$, let $t_x \in V(T)$ be the node of T_x that is the closest to s in T. For every node $t \in V(T)$, we set:

$$\chi_C(t) = \big(\chi(t) \cap V(H)\big) \cup \big\{x \in X \mid t \text{ belongs to the unique } (t_x, s)\text{-path in } T\big\}.$$

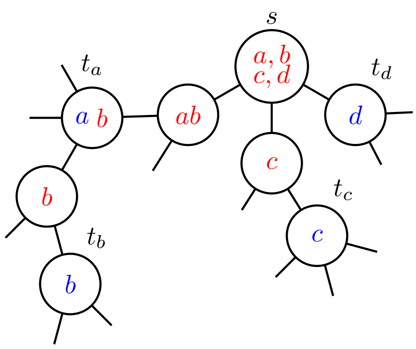

Figure 6: Suppose that $X = \{a, b, c, d\}$ and that node s has size $|\chi(s)| > k$. The nodes t_a, t_b, t_c and t_d are respectively the nodes of the traces T_a, T_b, T_c and T_d that are the closest to s. To construct $\mathcal{D}_C$, vertices a, b, c and d are respectively added to the the bag on the paths between s and t_a, t_b, t_C and t_d.

Finally T_C is obtained by restricting T to the nodes $V(T_C) = \{t \in V(T) \mid \chi_C(t) \neq \emptyset\}$. Observe that by construction of χ_C, for every vertex $x \in X \cup C$, the set $T_{C,x}$ $=\{t \in V(T_C) \mid x \in \chi_C(t)\}$ contains the trace T_x of x in $\mathcal{D}$ and the unique path in T between the trace T_x and the node s. It follows that $T_{C,x}$ is a connected subtree of T and so Lemma 7 applies. Consequently $\mathcal{D}_C = (T_C, \chi_C)$ of $H = G[X \cup C]$ is a loose tree-decomposition of H.

We now prove that $\mathcal{D}_C$ satisfies the statement of the claim. To that aim, we first prove that for every node $t \in V(T_C)$, $|\chi_C(t)| \leq |\chi(t)|$. As $\chi(s)$ and X both cover the tight bramble $\mathcal{B}$, by Lemma 4, any separator between X and $\chi(s)$ also covers $\mathcal{B}$. By the minimality of $|X|$, such a separator has size at least $|X|$. It follows by Menger Theorem that there exists a set $\mathcal{P}$ of $|X|$ vertex disjoint paths between X and $\chi(s)$. For every $x \in X$, we denote by P_x the path of $\mathcal{P}$ containing x. As (T, χ) is $\mathcal{B}'$-admissible and $|\chi(s)| \geq k+1$, $\chi(s)$ fails to cover $\mathcal{B}'$. Since $\chi(s)$ covers $\mathcal{B}$ and since $\mathcal{B}' = \mathcal{B} \cup \{C\}$, it follows that $\chi(s)$ does not intersect C. It follows that every path $P_x \in \mathcal{P}$ belongs to $G - C$ (see Figure 7). Let us now consider a node t for which there exists a vertex $x \in \chi_C(t) \setminus \chi(t)$. Observe that t lies in T_C on the unique (t_x, s)-path. By Lemma 2, $\chi(t)$ separates $\chi(t_x)$ and $\chi(s)$. It follows that $\chi(t)$ contains a vertex of P_x. As P_x is a path of $G - C$, for every vertex $x \in \chi_C(t) \setminus \chi(t)$, we identify a vertex $y \in \chi(t) \setminus \chi_C(t)$. It follows that $|\chi_C(t)| \leq |\chi(t)|$.

To conclude, let $t \in V(T_C)$ be a node such that $|\chi_C(t)| > k$. Since $|X| \leq k$ and since $\chi_C(t) \subseteq X \cup C$, we have that $\chi_C(t) \cap C \neq \emptyset$. As (T, χ) is $\mathcal{B}'$-admissible, $\chi(t)$ does not intersect some $B \in \mathcal{B}$. Let us

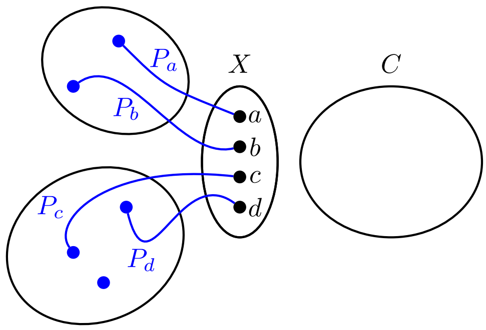

Figure 7: The set $\chi(s)$ is disjoint from C and $|X| \leq \chi(s)$. The paths P_a, P_b, P_c and P_d are pairwise vertex disjoint and do not intersect the component C.

show that $\chi_C(t)$ does not intersect B either. Assume it does. Then there exists $x \in B$ such that $x \in \chi_C(t) \setminus \chi(t)$. Observe that by construction of $\chi_C(t)$, $x \in X$ and thereby $\chi(t_x) \cap B \neq \emptyset$. As $\chi(s)$ covers $\mathcal{B}$, we also have that $\chi(s) \cap B \neq \emptyset$. By Lemma 2, $\chi(t)$ separates $\chi(s)$ and $\chi(t'_x)$. Since B is a connected subset of vertices, $\chi(t) \cap B \neq \emptyset$, a contradiction. Finally, as by construction every vertex of X belongs to $\chi_C(s)$, $\mathcal{D}_C$ is X-rooted. The claim follows $\diamond$

$\square$

3.3 Escape strategies derived from tight brambles

We now show how given a tight bramble of order k in a graph G, a fugitive can derive an escape strategy $(e_1, \mathbf{f}_G) \in \mathcal{F}_G$ such that for every search strategy $\mathbf{s}_G \in \mathcal{S}_G$ of cost k, the play $\mathcal{P}(\mathbf{s}_G, e_1, \mathbf{f}_G)$ generated by the program $\big(\mathbf{s}_G, (e_1, \mathbf{f}_G)\big)$ is infinite. In other words, the fugitive cannot be captured by a set of k searchers. This proves $(4 \Rightarrow 3)$ from Theorem 1.

Lemma 9. *Let G be a graph and $k \geq 1$ be an integer. If G has a tight bramble of order k, then* $\mathsf{avms}(G) \geq k$.

Proof. Let us first consider the case $k = 1$. By Observation 1, trees are the connected graphs for which the maximum order of a tight bramble is 1. Observe that, in a tree T, one searcher is enough to capture the fugitive. Suppose that the unique searcher is located at vertex x and that the fugitive is located at edge e. Then the search strategy consists in sliding along the unique edge xy incident to x towards e: that is, $\mathbf{s}_T(\{x\}, e) = y$ where y is the unique neighbor of x in the connected component of $T - x$ containing e. Clearly, eventually the fugitive will be located on an edge incident to a leaf of T and will be captured.

So let us consider $\mathcal{B}$ a tight bramble of G of order $k \geq 2$. Let $\mathbf{s}_G \in \mathcal{S}_G \subseteq \big(2^{V(G)}\big)^{\left(2^{V(G)} \times E(G)\right)}$ be an arbitrary search strategy such that $\mathsf{cost}(\mathbf{s}_G) < k$. Let B_0 be an arbitrary tight bramble element and let e_1 be an edge of $E(B_0)$. We build a fugitive strategy $(e_1, \mathbf{f}_G) \in \mathcal{F}_G$ such that the play $\mathcal{P}(\mathbf{s}_G, e_1, \mathbf{f}_G) = \langle S_0, e_1, S_1, \ldots, S_{i-1}, e_i, S_i, \ldots \rangle$ generated by the program $\big(\mathbf{s}_G, (e_1, \mathbf{f}_G)\big)$ verifies that for every $i \geq 1$, $\mathsf{fsp}_G(S_{i-1}, e_{i-1}, S_i) \neq \emptyset$. Such a fugitive strategy $(e_1, \mathbf{f}_G)$ certifies that the fugitive is never captured and thereby $\mathbf{s}_G$ is not winning. To that aim, we show that for every $i \geq 1$, there exists $B_i \in \mathcal{B}$ such that $S_i \cap B_i = \emptyset$ and $B_i \subseteq A_G(S_{i-1}, e_i, S_i)$. This allows us to select $e_{i+1} = \mathbf{f}_G(S_{i-1}, e_i, S_i) \in B_i$.

We proceed by induction on i. Clearly, since $k \geq 2$, the property is satisfied for $i = 1$. Indeed, as $S_0 = \emptyset$, we have that $|S_1| = 1$. Thereby S_1 is not a cover of $\mathcal{B}$. So there exists $B_1 \in \mathcal{B}$ which is disjoint from S_1. Suppose that $B_0 \neq B_1$ (otherwise, we are done). As B_0 and B_1 are touching connected subsets of vertices of G and as $|S_1| = 1$, we have that $B_1 \in A_G(S_0, e_1, S_1)$. It follows that we can set $\mathbf{f}_G(S_0, e_1, S_1) = e_2$ with e_2 being any edge of $E(B_1)$. Suppose that the property holds for every j such that $1 \leq j < i$. We distinguish three cases:

- $S_i = S_{i-1} \setminus \{u\}$ (that is, a searcher is removed from vertex u): It follows that $S_i \cap B_{i-1} = \emptyset$ and thereby we can set $B_i = B_{i-1}$ and $\mathbf{f}_G(S_{i-1}, e_i, S_i) = e_{i+1}$ with e_{i+1} being any edge of $E(B_i)$. Observe that the fugitive may not move at such a step.

- $S_i = S_{i-1} \cup \{v\}$ (that is, a new searcher is placed on vertex v): Observe that if $v \notin B_{i-1}$, then $B_{i-1} \cap S_i = \emptyset$. As $e_i \in E(B_{i-1})$ and B_{i-1} is connected, we also have that $B_{i-1} \subseteq A_G(S_{i-1}, e_i, S_i)$. Thereby we can set $B_i = B_{i-1}$ and $\mathbf{f}_G(S_{i-1}, e_i, S_i) = e_{i+1}$ with e_{i+1} being any edge of $E(B_i)$. So assume that $v \in B_{i-1}$. As $|S_i| < k$, S_i is not a cover of $\mathcal{B}$ and there exists $B_i \in \mathcal{B}$ such that $S_i \cap B_i = \emptyset$. Now observe that $S_i \cap B_{i-1} = \{v\}$. It follows that as B_{i-1} and B_i are touching, every edge in $E(B_i)$ is accessible through a (S_{i-1}, S_i)-avoiding pathway from e_i, that is $B_i \subseteq A_G(S_{i-1}, e_i, S_i)$. Thereby we can set $\mathbf{f}_G(S_{i-1}, e_i, S_i) = e_{i+1}$ with e_{i+1} being any edge of $E(B_i)$.

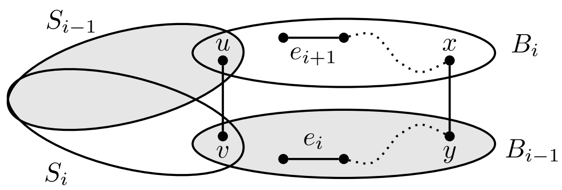

Figure 8: Two consecutive sets S_{i-1} and S_i of vertices occupied by the searchers from the play $\mathcal{P}$ such that $S_i \ominus S_{i-1} = \{u, v\} \in E(G)$ is an edge of multiplicity one. The tight ramble element $B_{i-1} \in \mathcal{B}$ that is disjoint from S_{i-1} contains the edge e_i. Likewise the tight bramble element $B_i \in \mathcal{B}$ that is disjoint from S_i contains the edge e_{i+1}. We observe that if B_{i-1} and B_i are disjoint, as they are touching, then there exists a pathway from e_i to e_{i+1} going through the edge xy that avoids the edge uv.

- $S_i \ominus S_{i-1} = \{u, v\}$ is an edge of $E(G)$ (that is, a searcher slides on the edge uv towards v): As in the previous case, observe that if $v \notin B_{i-1}$, then $B_{i-1} \cap S_i = \emptyset$. As $e_i \in E(B_{i-1})$ and B_{i-1} is connected, we also have that $B_{i-1} \subseteq A_G(S_{i-1}, e_i, S_i)$. Thereby we can set $B_i = B_{i-1}$ and $\mathbf{f}_G(S_{i-1}, e_i, S_i) = e_{i+1}$ with e_{i+1} being any edge of $E(B_i)$. So assume that $v \in B_{i-1}$. As $|S_i| < k$, S_i is not a cover of $\mathcal{B}$ implying the existence of $B_i \in \mathcal{B}$ such that $S_i \cap B_i = \emptyset$ (and so $v \notin B_i$). If $B_i \cap B_{i-1} \neq \emptyset$, then by the connectivity of the tight bramble elements, clearly every edge in $E(B_i)$ is accessible through a (S_{i-1}, S_i)-avoiding pathway from e_i. Suppose that $B_i \cap B_{i-1} = \emptyset$ (see Figure 8). As B_i and B_{i-1} are touching, there exists an edge xy distinct from $\{u, v\}$ such that $x \in B_i$ and $y \in B_{i-1}$. Again, the connectivity of the tight bramble elements implies that every edge in $E(B_i)$ is accessible through a (S_{i-1}, S_i)-avoiding pathway from e_i containing the edge xy. So in both cases, we have $B_i \subseteq A_G(S_{i-1}, e_i, S_i)$. Thereby we can set $\mathbf{f}_G(S_{i-1}, e_i, S_i) = e_{i+1}$ with e_{i+1} being any edge of $E(B_i)$. □

3.4 Monotone search strategies derived from loose tree-decompositions

We show that a loose tree-decomposition of width k for a graph G can be used to design a winning search strategy $\mathbf{s}_G \in \mathcal{S}_G$ of cost k that is monotone. This establishes $(2 \Rightarrow 5)$ from Theorem 1.

Lemma 10. *Let G be a graph and $k \geq 0$ be an integer. If $\mathsf{ctp}(G) \leq k$, then $\mathsf{mavms}(G) \leq k$.*

Proof. Suppose that $\mathsf{ctp}(G) \leq k$. By Lemma 6, G admits a full loose tree-decomposition $\mathcal{D} = (T, \chi)$ such that $\mathsf{width}(\mathcal{D}) \leq k$. We build from $\mathcal{D}$ a search strategy $\mathbf{s}_G \in \mathcal{S}_G \subseteq \left(2^{V(G)}\right)^{\left(2^{V(G)} \times E(G)\right)}$ and prove that it is a winning strategy of cost at most k. Consider an arbitrary fugitive strategy $(e_1, \mathbf{f}_G) \in \mathcal{F}_G$ and let $\mathcal{P}(\mathbf{s}_G, e_1, \mathbf{f}_G) = \langle S_0, e_1, S_1, \ldots, S_{i-1}, e_i, S_i, \ldots \rangle$ be the play generated by the program $\big(\mathbf{s}_G, (e_1, \mathbf{f}_G)\big)$. For $i \geq 1$, the searchers' position $S_i = \mathbf{s}_G(S_{i-1}, e_i)$ is defined as follows. At every step i, we guarantee the existence of a node $t \in V(T)$ such that $S_i \subseteq \chi(t)$. Pick a vertex $x \in V(G)$ and consider a node $t \in V(T)$ such that $x \in \chi(t)$. We set $S_1 = \{x\}$ and for $1 < i \leq \ell = |\chi(t)|$, $S_i = S_{i-1} \cup \{y\}$ for some

$y \in \chi(t) \setminus S_{i-1}$. That is, up to step ℓ, we iteratively add a searcher on every vertex of $\chi(t)$. Suppose that $i > \ell$. Let $t = V(T)$ be the node such that $S_{i-1} \subseteq \chi(t)$ and let $f = \{t_1, t_2\}$ be the tree-edge of T such that $e_i = \mathbf{f}_G(S_{i-2}, e_{i-1}, S_{i-1}) \in E(G[\chi(t_1) \cup \chi(t_2)])$. Suppose that t_1 is closer to t in T than t_2 is. Let t' be the neighbour of t in T such that t' and t_2 are in the same connected component of $T - tt'$. Observe that we may have $t = t_1$ and thereby $t' = t_2$. As $\mathcal{D}$ is full, there are $u \in \chi(t)$ and $v \in \chi(t')$ such that $\{u, v\} = \chi(t) \ominus \chi(t')$. There are two cases to consider:

1. if $uv \in E(G)$, then $S_i = \mathbf{s}_G(S_{i-1}, e_i) = \chi(t')$ is obtained by sliding along the edge uv (towards v);
2. if $uv \notin E(G)$, then $S_i = \mathbf{s}_G(S_{i-1}, e_i) = \chi(t) \setminus \{u\}$ is obtained by removing a searcher from vertex u, and $S_{i+1} = \mathbf{s}_G(S_i, e_{i+1}) = \chi(t')$, with $e_{i+1} = \mathbf{f}_G(S_{i-1}, e_i, S_i)$, is obtained by adding a searcher on vertex v.

Let us prove that $\mathbf{s}_G$ is a winning search strategy. To that aim, we show that there is some step $i \geq 1$ such that $\mathsf{fsp}_G(S_{i-1}, e_i, S_i) = \emptyset$. First observe, that if at step i a searcher is added, that is $S_i = S_{i-1} \cup \{u\}$ for some $u \in V(G)$, then $\mathsf{fsp}_G(S_{i-1}, e_i, S_i) \subsetneq \mathsf{fsp}_G(S_{i-2}, e_{i-1}, S_{i-1})$. Now suppose that at step i, a searcher slides on the edge uv where $\{u, v\} = \chi(t) \ominus \chi(t')$. By Lemma 2, $\chi(t)$ is a separator of G. Observe that, by construction of $\mathbf{s}_G$, either $e_i = uv$ or e_i and v belongs to the same component of $G - \chi(t)$. It follows that $\mathsf{fsp}_G(S_{i-1}, e_i, S_i) \subsetneq \mathsf{fsp}_G(S_{i-2}, e_{i-1}, S_{i-1})$. Finally, suppose that at step i, a searcher is removed from a vertex $u \in S_{i-1}$. By construction of $\mathbf{s}_G$, this is only the case where the set $\{u, v\} = \chi(t) \ominus \chi(t')$ does not correspond to an edge of $E(G)$. By Lemma 3, $\chi(t) \cap \chi(t')$ is a separator of G. Observe in that case that $\mathsf{fsp}_G(S_{i-1}, e_i, S_i) = \mathsf{fsp}_G(S_{i-2}, e_{i-1}, S_{i-1})$. But as, by construction, $\mathbf{s}_G$ does not contain two consecutive steps removing a searcher, we can conclude that there exists some step i such that $\mathsf{fsp}_G(S_{i-1}, e_i, S_i) = \emptyset$.

Moreover observe that, from the discussion above, for every $(e_1, \mathbf{f}_G) \in \mathcal{F}_G$, the play generated by the program $\big(\mathbf{s}_G, (e_1, \mathbf{f}_G)\big)$ is monotone. This, in turns, implies that the strategy $\mathbf{s}_G$ is monotone. Finally we observe that by the construction of $\mathbf{s}_G$, we have $\mathsf{cost}(\mathbf{s}_G) = \mathsf{ctp}(G) \leq k$, proving the result. ☐

4 Discussion

In this paper, we defined the concepts of loose tree-decomposition and tight bramble. We showed that the existence of a tight bramble of large order in a graph is an obstacle to a loose tree-decomposition of small width. Moreover, we proved the equivalence of the corresponding parameters to the Cartesian product number, introduced as the *largeur d'arboresence* by Colin de Verdière [22] (see also [10]) and the mixed search number against an agile and visible fugitive (Theorem 1).

One may consider path-like counterparts of all these concepts as follows. We can restrict the definition of loose tree-decomposition to path instead of tree, then we obtain *loose path-decompositions*. Similarly, if in the definition of Cartesian tree product number, we again consider trees instead of paths, this yields the definiton of the *Cartesian path product number*. A "path analogue" of Theorem 1 may easily be produced by considering the *mixed search number against an agile and invisible fugitive* introduced by Bienstock and Seymour in [4], while the min-max analogue for these path-like parameters can be derived by the framework of [7]. Finally, we wish to mention that a different variant for path-decomposition that is equivalent the above parameters was introduced by Takahashi and Ueno and Kajitani in [21, 20] under the name *proper path decomposition*.

Finally, our results may support the belief that it might be possible to derive the min-max duality of Theorem 1 using general parameter duality frameworks such as those in [16, 13, 1, 14].

Acknowledgments. We would like to thank the anonymous referees for their careful reading and accurate remarks that helped us to greatly improve the presentation of the manuscript.

References

[1] Omid Amini, Frédéric Mazoit, Nicolas Nisse, and Stéphan Thomassé. Submodular partition functions. *Discret. Math.*, 309(20):6000–6008, 2009. doi:10.1016/j.disc.2009.04.033.

[2] Patrick Bellenbaum and Reinhard Diestel. Two short proofs concerning tree-decompositions. *Combinatorics, Probability and Computing*, 11(6):541–547, 2002. doi:10.1017/S0963548302005369.

[3] Daniel Bienstock, Niel Robserston, Paul D. Seymour, and Robin Thomas. Quickly excluding a forest. *Journal of Combinatorial Theory, Series B*, 52:274–283, 1991. doi:10.1016/0095-8956(91)90068-U.

[4] Daniel Bienstock and Paul D. Seymour. Monotonicity in graph searching. *Journal of Algorithms*, 12(2):239–245, 1991. doi:10.1016/0196-6774(91)90003-H.

[5] Nick D. Dendris, Lefteris M. Kirousis, and Dimitrios M. Thilikos. Fugitive-search games on graphs and related parameters. *Theoretical Computer Science*, 172:233–254, 1997. doi:10.1016/S0304-3975(96)00177-6.

[6] Jonathan A. Ellis, Ivan Hal Sudborough, and Jonathan S. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113(1):50–79, 1994. doi:10.1006/inco.1994.1064.

[7] Fedor V. Fomin and Dimitrios M. Thilikos. On the monotonicity of games generated by symmetric submodular functions. *Discret. Appl. Math.*, 131(2):323–335, 2003. doi:10.1016/S0166-218X(02)00459-6.

[8] Fedor V. Fomin and Dimitrios M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399:236–245, 2008. doi:10.1016/j.tcs.2008.02.040.

[9] Rudolph Halin. S-functions for graphs. *Journal of Geometry*, 8:171–186, 1976. doi:10.1007/BF01917434.

[10] Daniel J. Harvey. *On treewidth and graph minors.* PhD thesis, University of Melbourne, 2014.

[11] Nancy G. Kinnersley. The vertex separation number of a graph equals its path-width. *Information Processing Letters*, 42(6):345–350, 1992. doi:10.1016/0020-0190(92)90234-M.

[12] Lefteris M. Kirousis and Christos H. Papadimitriou. Searching and pebbling. *Theoretical Computer Science*, 47(2):205–218, 1986. doi:10.1016/0304-3975(86)90146-5.

[13] Laurent Lyaudet, Frédéric Mazoit, and Stéphan Thomassé. Partitions versus sets: A case of duality. *Eur. J. Comb.*, 31(3):681–687, 2010. doi:10.1016/j.ejc.2009.09.004.

[14] Frédéric Mazoit and Nicolas Nisse. Monotonicity of non-deterministic graph searching. *Theor. Comput. Sci.*, 399(3):169–178, 2008. doi:10.1016/j.tcs.2008.02.036.

[15] Nicolas Nisse. Network decontamination. In Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors, *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*, pages 516–548. Springer, 2019. doi:10.1007/978-3-030-11072-7_19.

[16] Ronan Pardo Soares. *Pursuit-Evasion, Decompositions and Convexity on Graphs.* Theses, Université Nice Sophia Antipolis, November 2013. URL: https://tel.archives-ouvertes.fr/tel-00908227.

[17] Torrence D. Parsons. Pursuit-evasion in a graph. In *Theory and applications of graphs*, volume 642 of *Lecture Nots in Mathematics*, pages 426–441, 1978. doi:10.1007/BFb0070400.

[18] Niel Roberston and Paul D. Seymour. Graph minors. III. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36:49–64, 1984. doi:10.1016/0095-8956(84)90013-3.

[19] Paul D. Seymour and Robin Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33, 1993. doi:10.1006/jctb.1993.1027.

[20] Atsushi Takahashi, Shuichi Ueno, and Yoji Kajitani. Minimal acyclic forbidden minors for the family of graphs with bounded path-width. *Discret. Math.*, 127(1-3):293–304, 1994. `doi:10.1016/0012-365X(94)90092-2`.

[21] Atsushi Takahashi, Shuichi Ueno, and Yoji Kajitani. Mixed searching and proper-path-width. *Theoretical Computer Science*, 137(2):253–268, 1995. `doi:10.1016/0304-3975(94)00160-K`.

[22] Yves Colin De Verdière. Multiplicities of eigenvalues and tree-width of graphs. *Journal of Combinatorial Theory, Series B*, 74(2):121–146, 1998. `doi:10.1006/jctb.1998.1834`.