Theoretical Computer Science 463 (2012) 96–113

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Approximate search strategies for weighted trees

Dariusz Dereniowski *

Department of Algorithms and System Modeling, Gdańsk University of Technology, Gdańsk, Poland

ARTICLE INFO

Keywords:
Connected searching
Edge searching
Graph searching
Node searching
Pathwidth

ABSTRACT

The problems of (classical) searching and connected searching of weighted trees are known to be computationally hard. In this work we give a polynomial-time 3-approximation algorithm that finds a connected search strategy of a given weighted tree. This in particular yields constant factor approximation algorithms for the (non-connected) classical searching problems and for the weighted pathwidth problem for this class of graphs.

1. Introduction

Consider the following game (known as the *edge searching* problem) between a fugitive that is hidden in the graph and a team of searchers that needs to capture the fugitive. The fugitive is considered captured if a searcher shares its location. The fugitive is invisible which implies that the searchers can deduce its location only on the basis of the history of their moves. In each move a single searcher can either be removed from its current vertex, placed on a vertex (if the searcher does not occupy a vertex) or can slide along an edge from its current vertex to one of its neighbors. The fugitive is fast, which implies that it can traverse, between two consecutive moves of the searchers, a path of arbitrary length that is free of searchers. Finally, the fugitive has the complete knowledge about the graph and about the moves (including the ones to come) of the searchers. The latter implies that it will avoid being captured as long as possible. Thus, since the searchers need to analyze the worst case scenario, this is a one player game where the goal is to design a search strategy (a sequence of moves of the searchers) that uses the given number of searchers and guarantees the capture of the fugitive, or to decide that no such search strategy exists. It turns out that it is not beneficial to allow recontamination of an edge that has been previously cleared (i.e. an edge that was guaranteed not to contain the fugitive), or in other words, we say that the edge searching problem is monotone [26].

In this work we are interested in a modification of the above game called *connected searching*, where one has to ensure that the subgraph induced by the clear edges is always connected. It turns out that the connected searching problem is not monotone [33]. Our interest focuses on a generalization of this problem where the graph to be cleared is weighted. In this setting the weight $w(v)$ of a vertex v determines the number of searchers needed to guard it, i.e. if less than $w(v)$ searchers occupy v, then the fugitive is able to pass this vertex without being captured. Similarly, the weight $w(e)$ of an edge e determines the minimum number of searchers that have to simultaneously slide along e to clear it. In this work we restrict our attention to weighted trees. It is known that the connected searching problem is monotone for such graphs [2]. The monotonicity allows us to simplify our description of the problem, namely all searchers are forced to be initially placed on the same vertex, called the *homebase*, and the only allowed moves are sliding along the edges. If each sliding move clears an edge and, as a result of the move, no edge becomes recontaminated, then these assumptions guarantee that the search strategy is connected. Note that the choice of the homebase may affect the minimum number of searchers that have to be used to clear the graph.

* Tel.: +48 583471956; fax: +48 583471766.
E-mail address: deren@eti.pg.gda.pl.

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.07.006

1.1. Related work

The edge searching problem is closely related to pathwidth, namely the difference between the pathwidth of a simple graph and the minimum number of searchers needed to clear it is at most one [4,22–24,29]. The pathwidth problem can be solved efficiently for k-starlike graphs [20], circular-arc graphs [32], graph of bounded treewidth [8], trees [15,27], unicyclic graphs [14], permutation graphs [9], block graphs [11], or cographs [10]. On the other hand, the pathwidth and the graph searching problems are NP-complete for chordal graphs [20] or bipartite distance hereditary graphs [25]. There exists a $O(\log^2 n)$-approximation polynomial-time algorithm for finding path decompositions of general graphs [7], where n is the number of vertices of a graph. The approximation algorithm presented in [16] that finds, for a given graph of treewidth k, a tree decomposition of width $O(k\sqrt{\log k})$ gives a $O(\sqrt{\log k}\log n)$-approximation algorithm for finding minimum width path decompositions which improves the previously mentioned approximation ratio of $O(\log^2 n)$. For approximate solutions for special classes of graphs see e.g. [6,18]. For some connections between graph searching problems and width-like parameters see also [5,7,19,21,31].

The ratio between the minimum number of searchers needed to clear the graph in a connected way (or connected pathwidth) and the number of searchers needed to clear the graph in a classical way (or the pathwidth) is asymptotically 2 [13] (we refer here both to node searching and edge searching). Moreover, [13] gives an efficient algorithm that converts a search strategy using k searchers into a connected search strategy that uses at most $2k + 3$ searchers. Thus, the algorithm described in [16] and the above correspondences between the pathwidth and the search number give an $O(\sqrt{\log k}\log n)$-approximation algorithm for finding connected search strategies of any simple graph G, where k equals the treewidth of G. See also [1,3,17,30] for results and discussions on the relation between the classical and connected searching numbers.

Much less is known when one considers the above problems for weighted graphs. The connected searching and the pathwidth problems are NP-complete for weighted trees [12,28]. As observed in [28], the pathwidth of a weighted graph G equals the pathwidth of the (unweighted) graph G' obtained from G by replacing each vertex v of G with $w(v)$ vertices inducing a complete subgraph in G', where $w(v)$ is the weight of v in G, and two vertices in different cliques of G' are adjacent if and only if they belong to complete subgraphs that correspond to two adjacent vertices in G. Thus, the equivalence between the pathwidth and the node searching problems carries over to weighted graphs. One of the advantages of the above correspondences between the weighted pathwidth and the weighted searching problems is that a $O(q)$-approximation algorithm for any of them yields a $O(q)$-approximation algorithm for the other.

1.2. Our results

This work is a self-contained continuation of [12]. We give a 3-approximation algorithm for connected searching of weighted trees, which yields a constant factor approximation algorithm for pathwidth of weighted trees. The next section recalls a formal definition of the searching problem, while Section 3 gives the background properties of connected search strategies in weighted trees. In particular, Section 3.1 introduces the notion of a greedy search strategy, which is the main tool in our analysis. The main difficulty in finding an optimal connected search strategy of a weighted tree lies in finding an appropriate permutation of clearing the edges leading from each vertex to its neighbors, i.e. informally speaking, finding this permutation is as difficult as finding an optimal connected search strategy itself [12]. Section 4 gives a description and an analysis of a generic algorithm whose input consists of a weighted tree T and a permutation set Γ. The permutation set provides for each vertex v a set of permutations of the edges leading from v to its children (with respect to all possible choices of the root). The algorithm returns a connected search strategy that for each vertex v of T uses one of the permutations in Γ provided for v. The generic algorithm is described in Section 4.1 and we prove in Section 4.2 that it finds the best, i.e. using the minimum number of searchers, search strategy that follows the above restriction. Then, Section 4.3 deals with the complexity of the algorithm and it is proven that its running time is polynomial in the size of the tree and in the size of Γ. The above is a good point of departure for the analysis of approximate solutions, because in order to prove that there exists a k-approximation polynomial-time algorithm it is enough to define a polynomially bounded permutation set Γ for the input tree T and to prove that there exists a connected search strategy that respects the clearing order of the edges given by Γ and the number of searchers it uses does not exceed k times the minimum number of searchers needed for T. Section 5 takes this route by providing a simple permutation set Γ_1 for which the generic algorithm is guaranteed to achieve the ratio of $k = 3$. Informally speaking, Γ_1 is constructed so that for each $v \in V(T)$, for each choice of the root for T and for each descendant u of v there is exactly one permutation such that uv is ordered last among the edges between v and its descendants (the edges leading to other descendants are ordered arbitrarily by this permutation).

2. Preliminaries

First we describe the edge searching problem for weighted graphs $G = (V(G), E(G), w)$, where $w\colon V(G) \cup E(G) \to \mathbb{N}_+$, and let us start by describing the properties of the fugitive. The fugitive can occupy the vertices and the edges of G and it can change the location at any moment (i.e. between two subsequent moves of the searchers) by traversing any path such that each vertex v of the path is occupied by less than $w(v)$ searchers. We say that an edge (a vertex) is *clear* if it cannot contain the fugitive. Otherwise the edge or the vertex is said to be *contaminated*. (Note that if an edge is clear, then, by construction,

both of its endpoints are also clear.) The *clear subgraph* in a particular point of time is the subgraph composed of the clear edges and the clear vertices of G. The fugitive is captured when either:

- it is located on an edge e of G and at least $w(e)$ searchers simultaneously slide along e, or
- it is located on a vertex v of G and at least $w(v)$ searchers occupy v.

Also, the fugitive is invisible and has the complete knowledge about the graph, the locations of the searchers and their future moves. The latter in particular means that he will avoid being captured as long as possible.

Given a weighted graph $G = (V(G), E(G), w)$, a *search strategy* $\mathcal{S}$ for G is a sequence $\mathcal{S}[1], \ldots, \mathcal{S}[l]$ of three types of moves:

(a) placing s searchers on a vertex, $s \geq 1$,
(b) removing s searchers from a vertex $v \in V(G)$, $s \geq 1$, provided that at least s searchers are present at v at the end of the preceding move,
(c) sliding s searchers from u to v, $u, v \in V(G)$, $uv \in E(G)$, $s \geq 1$, provided that at least s searchers are present at u at the end of the preceding move.

We say that $\mathcal{S}$ is a *k-search strategy* for G, where k is an integer, if $\mathcal{S}$ is a search strategy for G and uses at most k searchers.

A search strategy $\mathcal{S}$ is *partial* if a subset of the edges of G is clear at the end of $\mathcal{S}$. (Thus, a partial search strategy may, but does not have to clear all edges of G.) For brevity let $|\mathcal{S}|$ be the number of moves in $\mathcal{S}$. We say that $\mathcal{S}$ *clears* a subgraph G' of G if all edges and all vertices of G' are clear when $\mathcal{S}$ is finished, i.e. at the end of the move $\mathcal{S}[|\mathcal{S}|]$. Note that if $v' \in V(G')$ has a neighbor $v \in V(G) \setminus V(G')$ in G, then at least $w(v')$ searchers are present at v' at the end of the move $\mathcal{S}[|\mathcal{S}|]$ to protect G' from recontamination. Denote by $|\mathcal{S}[i]|$, $i \in \{1, \ldots, |\mathcal{S}|\}$, the number of searchers used in step $\mathcal{S}[i]$. Let $C_E(\mathcal{S})$ and $C_V(\mathcal{S})$ be, respectively, the sets of edges and vertices cleared by $\mathcal{S}$. Note that if $\mathcal{S}$ clears G, then $C_E(\mathcal{S}) = E(G)$ and $C_V(\mathcal{S}) = V(G)$. Define $\mathcal{S}[\preceq i]$, $i \in \{1, \ldots, |\mathcal{S}|\}$, to be the partial search strategy obtained by performing the first i moves of $\mathcal{S}$. We say that a vertex v is *guarded* at the end of a move $\mathcal{S}[i]$ if v is occupied by at least $w(v)$ searchers at the end of $\mathcal{S}[i]$ and at least one edge incident to v is contaminated at the end of $\mathcal{S}[i]$. (Thus, if at least $w(v)$ searchers occupy v and all edges incident to v are clear at the end of $\mathcal{S}[i]$, then we do not classify v as guarded.) Let $\delta(\mathcal{S})$ be the set of vertices guarded at the end of $\mathcal{S}$. If $\mathcal{S}$ clears G, then clearly $\delta(\mathcal{S}) = \emptyset$. Using the above notation, $\delta(\mathcal{S}[\preceq i])$, $i \in \{1, \ldots, |\mathcal{S}|\}$, is the set of the vertices guarded at the end of $\mathcal{S}[i]$, i.e. at the end of the i-th move of $\mathcal{S}$.

A graph is *connected* if there exists a path between any pair of its vertices. We say that a search strategy $\mathcal{S}$ is *connected* if the subgraph cleared by $\mathcal{S}[\preceq i]$ is connected for each $i = 1, \ldots, |\mathcal{S}|$. In the remaining part of this work we assume for brevity that each search strategy that is partial is also connected. A partial search strategy $\mathcal{S}$ for G can be *k-extended* to a partial search strategy $\mathcal{S}'$ for G if $|\mathcal{S}'| \geq |\mathcal{S}|$, $\mathcal{S}[i] = \mathcal{S}'[i]$ for each $i = 1, \ldots, |\mathcal{S}|$ and $|\mathcal{S}'[i]| \leq k$ for each $i = |\mathcal{S}| + 1, \ldots, |\mathcal{S}'|$. In such case we also say that $\mathcal{S}'$ is a *k-extension*, or an *extension* for short, of $\mathcal{S}$.

Proposition 1. *Given a weighted graph G and an integer k, if there exists a partial k-search strategy that clears the edges in $X \subseteq E(G)$, then there exists a partial k-search strategy $\mathcal{S}$ such that $C_E(\mathcal{S}) = X$, initially k searchers occupy the same vertex of G and $\mathcal{S}[1], \ldots, \mathcal{S}[|\mathcal{S}|]$ are sliding moves.* □

Due to the above, we simplify the notation by assuming that initially all searchers are placed on a selected vertex v of the given graph G and $\mathcal{S}[1], \ldots, \mathcal{S}[|\mathcal{S}|]$ are sliding moves. The vertex v is called the *homebase* of $\mathcal{S}$.

We use the symbol $\mathtt{s}(\mathcal{S})$ to denote the number of searchers used by $\mathcal{S}$, $\mathtt{s}(\mathcal{S}) = \max\{|\mathcal{S}[i]| : i = 1, \ldots, |\mathcal{S}|\}$. Then, $\mathtt{s}(G)$ is the *search number* of G and equals the minimum number k such that there exists a k-search strategy for G. The *connected search number* of G, denoted by $\mathtt{cs}(G)$, is the minimum integer k such that there exists a connected k-search strategy for G. A (connected) search strategy is *optimal* if it uses $\mathtt{s}(G)$ (respectively, $\mathtt{cs}(G)$) searchers. We extend this notation for the case when the homebase of a search strategy is given in advance. Thus, $\mathtt{cs}(G, r)$, $r \in V(G)$, is the minimum number of searchers k such that there exists a connected k-search strategy with homebase r. If $X \subseteq V(G) \cup E(G)$, then we write for brevity $w(X) = \sum_{x \in X} w(x)$.

Suppose that we select $w(v)$ searchers that occupy each vertex v that needs to be guarded at the end of a particular move of a search strategy. Then, the remaining searchers, i.e. the ones that have not been selected, are called *free*. Note that any subset of size $w(v)$ of the searchers present at v can be selected to determine the free searchers, and to avoid ambiguity one can assume that the searchers have unique identifies and the selection is performed by taking the searchers with the smallest $w(v)$ identifiers.

Now we recall some basic graph-theoretic notation. A *tree* $T = (V(T), E(T))$ is an acyclic connected graph. Given a (not necessarily connected) graph, each of its maximal connected subgraphs is called a *connected component*. Unless stated otherwise, each tree is rooted at a vertex r. For $v \in V(T)$ we define E_v and V_v to be the set of edges between v and its children and the set of the children of v, respectively. Given $X \subseteq V(T)$, we say that T' is a subgraph of T *induced* by X if $V(T') = X$ and $E(T') = \{uv \in E(T) : u, v \in X\}$. Given a tree T rooted at r and a vertex $v \in V(T)$, T_v is the subtree of T rooted at v and induced by v and all its descendants in T. For $X \subseteq V(T)$ or for $X \subseteq E(T)$ we write $T - X$ to denote the (not necessarily connected) subgraph of T obtained by removing all vertices in X or, respectively, all edges from T. Given a tree T and $v \in V(T)$, $N_T(v)$ denotes the set of neighbors of v in T. Thus, if T is rooted, then $N_T(v)$ consists of the children of v and, if v is not the root of T, the parent of v. In the following each partial search strategy for a rooted tree has the property that the roof of the tree is the homebase of the search strategy.

While solving the graph searching problem in a certain class of graphs, one has to address the question concerning the monotonicity: is it beneficial to consider a search strategy that allows the fugitive to reach an edge or a vertex that has been previously cleared? The answer gives the following.

Theorem 1 (*[2]*). *For each weighted tree T, there exists a connected $\mathtt{cs}(T)$-search strategy in which no recontamination is possible.* □

Due to the above, in the remainder of this paper each search strategy we consider does not allow for recontamination.

Now we give some comments on the structure of each connected search strategy $\mathcal{S}$ for a weighted tree T. Consider two moves $\mathcal{S}[i]$ and $\mathcal{S}[j]$, $i < j$, that clear two edges e_1, e_2 of T, respectively, and all the intermediate moves $\mathcal{S}[i+1], \ldots, \mathcal{S}[j-1]$ do not clear any edges of T (thus the searchers slide along the clear edges). Without loss of generality, the intermediate moves are needed to gather the sufficient number of searchers at an endpoint of e_2 in order to clear the edge. Since the searchers used for guarding at the end of $\mathcal{S}[i]$ cannot change their locations during the intermediate moves, the free searchers move towards the endpoint of e_2. Moreover, any subset (of the appropriate size) of the searchers can be used to clear e_2. Thus, in the following we list only the clearing moves of each search strategy we consider. The above, Proposition 1 and Theorem 1 imply in particular that $|\mathcal{S}| = |E(T)|$ for each connected search strategy $\mathcal{S}$ for T.

As observed in [12], we can make some further simplifying assumptions. Let T be a weighted tree rooted at r. First, note that if for an edge uv, where u is the parent of v it holds $w(v) \geq w(uv)$, then changing the weight of uv to be 1 does not affect $\mathtt{cs}(T, r)$, because in either case at least $w(v)$ searchers need to slide along uv to clear it. Moreover, we can subdivide each edge uv of T with $w(uv) > w(v)$ (i.e. replace uv with a path consisting of two edges so that the endpoints of the path are u and v in T) and make the new internal vertex to have the weight $w(uv)$ and the two new edges to have weights 1. It is not difficult to prove that the problems of connected searching for trees before and after this transformation are equivalent, as long as r is the homebase in both cases.

Note that the extension of a tree T by adding a child of weight 1 to each leaf of T does not change its connected search number. However, the new tree T' has the property that the number of searchers used during an i-th move of a connected search strategy $\mathcal{S}$, i.e. while sliding some searchers along an edge, does not exceed the number of searchers used for guarding at the beginning or at the end of the i-th move. Indeed, if the searchers that are sliding reach a vertex v that has some descendants, then (due to the earlier simplifying assumptions) they all need to occupy the vertex they arrived at. On the other hand, if the vertex has no descendants, then its weight is 1 and, due to the construction of T', it is the only child of its parent u, which means that one searcher that was used for guarding at the end of $(i-1)$-st move of $\mathcal{S}$ can move from u to v to clear uv, and therefore, no additional searchers are needed while clearing uv. The following proposition summarizes the above discussion.

Proposition 2. *For each weighted tree $T = (V(T), E(T), w)$ rooted at r there exists a weighted tree $T' = (V(T'), E(T'), w')$ rooted at r' that satisfies the following conditions:*

- *$w'(e) = 1$ for each $e \in E(T')$,*
- *$|V(T')| \leq 2|V(T)|$ and $|E(T')| \leq 2|E(T)|$,*
- *if $\mathcal{S}'$ is a connected search strategy for T', then $\mathtt{s}(\mathcal{S}') = \max\{k, \max\{w(\delta(\mathcal{S}'[\preceq i])) : i = 1, \ldots, |\mathcal{S}'|\}\}$, where k is the number of searchers initially placed by $\mathcal{S}'$ on r', and*
- *$\mathtt{cs}(T, r) = \mathtt{cs}(T', r')$.*

Moreover, the tree T' can be computed in time $O(|V(T)|)$ and for a given connected search strategy $\mathcal{S}'$ for T' with homebase r', a connected $\mathtt{s}(\mathcal{S}')$-search strategy for T with homebase r can be computed in time $O(|V(T)|)$. □

The above allows us to assume in the remaining parts of this paper that T is a node-weighted tree and for any partial search strategy $\mathcal{S}$ for T it holds

$$\mathtt{s}(\mathcal{S}) = \max\{k, \max\{w(\delta(\mathcal{S}[\preceq i])) : i = 1, \ldots, |\mathcal{S}|\}\}, \tag{1}$$

where k is the number of searchers initially placed on the homebase of $\mathcal{S}$.

We use the symbol $\varepsilon\langle T, k\rangle$ to denote a search strategy that places k searchers on the root of T, but does not clear any edges. Moreover, $\varepsilon\langle T\rangle$ denotes the set of all search strategies of this type. In the following $\langle u, v\rangle$, $uv \in E(T)$, denotes a move in which exactly $w(v)$ searchers slide from u to v in T.

3. Connected searching of weighted trees — basic properties

This section introduces the basic facts used in the algorithm described in Section 4. Some of the ideas that are present in an optimal algorithm for connected searching of bounded degree weighted trees [12] proved to be useful for finding approximate solution for the general case presented in this work. However, the concept of *minimal* search strategies used in [12] has been replaced by the idea of *greedy* search strategies, described in Section 3.1. The advantages are a simpler analysis and reduced running time of the algorithm.

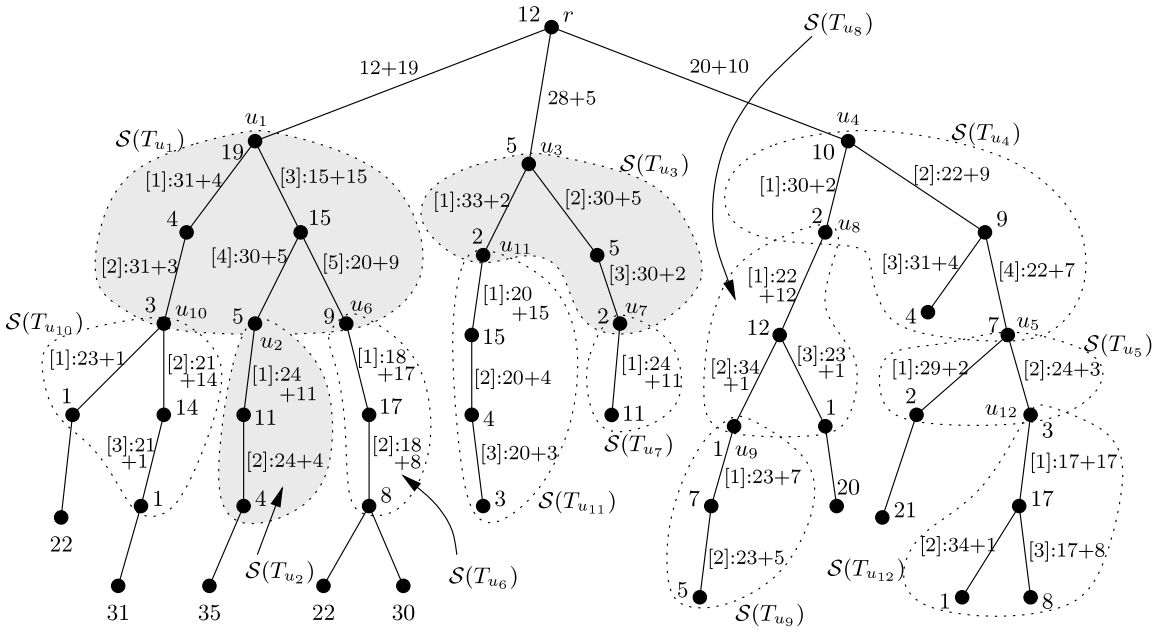

Fig. 1. A 35-search strategy $\langle r, u_1\rangle \oplus \mathcal{S}(T_{u_1}) \oplus \mathcal{S}(T_{u_2}) \oplus \langle r, u_3\rangle \oplus \mathcal{S}(T_{u_3}) \oplus \langle r, u_4\rangle \oplus \mathcal{S}(T_{u_4}) \oplus \cdots \oplus \mathcal{S}(T_{u_{12}})$.

Now we define a binary operator $\oplus$, which can be seen as a 'concatenation' of two search strategies. Let $\mathcal{S}_1$ be a partial connected search strategy for T with homebase r and let $\mathcal{S}_2$ be a partial connected search strategy for a connected component T' in $T - C_E(\mathcal{S}_1)$ with homebase $v \in \delta(\mathcal{S}_1)$. Then, $\mathcal{S}_1 \oplus \mathcal{S}_2$ is a search strategy with homebase r defined as:

(i) $(\mathcal{S}_1 \oplus \mathcal{S}_2)[i] = \mathcal{S}_1[i]$ and $\delta((\mathcal{S}_1 \oplus \mathcal{S}_2)[\preceq i]) = \delta(\mathcal{S}_1[\preceq i])$ for each $i = 1, \ldots, |\mathcal{S}_1|$, and
(ii) the move $(\mathcal{S}_1 \oplus \mathcal{S}_2)[|\mathcal{S}_1| + i]$ clears the same edge as $\mathcal{S}_2[i]$ and the set of guarded vertices at the end of this move is $\delta((\mathcal{S}_1 \oplus \mathcal{S}_2)[|\mathcal{S}| + i]) = (\delta(\mathcal{S}_1) \setminus \{v\}) \cup \delta(\mathcal{S}_2[\preceq i])$ for each $i = 1, \ldots, |\mathcal{S}_2|$.

Note that by the definition, $\delta(\mathcal{S}_2[\preceq i]) \subseteq V(T')$ for each $i = 1, \ldots, |\mathcal{S}_2|$ and if all edges of T' incident to v have been cleared by $\mathcal{S}_2[\preceq i]$, then $v \notin \delta(\mathcal{S}_2[\preceq i])$.

Now we give a remark about computing the number of searchers that $\mathcal{S}_1 \oplus \mathcal{S}_2$ uses. Clearly, $\mathtt{s}(\mathcal{S}_1)$ searchers are needed to perform the first $|\mathcal{S}_1|$ moves of $\mathcal{S}_1 \oplus \mathcal{S}_2$. Then, the edges in $C_E(\mathcal{S}_2)$ are cleared by $\mathcal{S}_2$, which requires $\mathtt{s}(\mathcal{S}_2)$ searchers, while the vertices in $\delta(\mathcal{S}_1) \setminus \{v\}$ need to be guarded. The vertex v is guarded at the end of $\mathcal{S}_1$ and this may change during the execution of the moves of $\mathcal{S}_2$ in $\mathcal{S}_1 \oplus \mathcal{S}_2$. Hence we obtain the following.

Lemma 1. *Let T be a weighted rooted tree. If $\mathcal{S}_1$ is a partial connected search strategy for T and $\mathcal{S}_2$ is a partial connected search strategy for a connected component in $T - C_E(\mathcal{S})$ with homebase v, where $v \in \delta(\mathcal{S}_1)$, then $\mathtt{s}(\mathcal{S}_1 \oplus \mathcal{S}_2) = \max\{\mathtt{s}(\mathcal{S}_1), w(\delta(\mathcal{S}_1) \setminus \{v\}) + \mathtt{s}(\mathcal{S}_2)\}$.* □

Fig. 1 gives an example of a partial search strategy. To simplify the presentation, this tree T does not have leaves of weights 1 attached as justified at the end of Section 2. Each subtree $T' \subseteq T_u$, $u \in V(T)$, distinguished with a 'dotted' curve is cleared by the corresponding connected search strategy denoted by $\mathcal{S}(T_u)$. The search strategy is described by giving the order of clearing the edges of T' and the number of searchers used in a particular move. The edges of T' have labels of the form $[i] : x + y$, which we interpret as follows: the i-th move of the corresponding search strategy $\mathcal{S}(T_u)$ for T', i.e. the move $\mathcal{S}(T_u)[i]$, uses x searchers for guarding and y searchers slide along the particular edge (due to the choice of the homebase r, sliding always occurs towards the leaves of T).

Fig. 1 presents the following partial 35-search strategy with homebase r:

$$\mathcal{S} = \langle r, u_1\rangle \oplus \mathcal{S}(T_{u_1}) \oplus \mathcal{S}(T_{u_2}) \oplus \langle r, u_3\rangle \oplus \mathcal{S}(T_{u_3}) \oplus \langle r, u_4\rangle \oplus \mathcal{S}(T_{u_4}) \oplus \cdots \oplus \mathcal{S}(T_{u_{12}}).$$

For example, $\mathcal{S}(T_{u_1})$, shown in Fig. 1, consists of five clearing moves $\mathcal{S}(T_{u_1})[1], \ldots, \mathcal{S}(T_{u_1})[5]$, where $\mathcal{S}(T_{u_1})[1]$ slides 4 searchers from u_1 to its left child while 31 searchers are used for guarding (during this move r and u_1 need to be guarded, $w(r) + w(u_1) = 31$); in $\mathcal{S}(T_{u_1})[2]$ we have that 3 searchers reach u_{10} (while r and u_1 are guarded); in $\mathcal{S}(T_{u_1})[3]$ we have that 15 searchers slide from u_1 to its right child (during this move r and u_{10} need to be guarded); in $\mathcal{S}(T_{u_1})[4]$ we have that 5 searchers reach u_2 and 30 searchers are used to guard r, u_{10} and the right child of u_1; finally $\mathcal{S}(T_{u_1})[5]$ slides 9 searchers that reach u_6 and simultaneously guards r, u_2 and u_{10}. Note that $\mathcal{S}$ can be extended to a 35-search strategy for T (the remaining edges in $E(T) \setminus C_E(\mathcal{S})$ need to be cleared in an unique order). We point out two facts that are illustrated in this example. First, if the partial search strategy $\langle r, u_1\rangle \oplus \mathcal{S}(T_{u_1}) \oplus \mathcal{S}(T_{u_2})$ is extended with clearing any edge of T_{u_1}, then this new search strategy cannot be extended to a partial 35-search strategy that clears an edge of T_{u_3} or of T_{u_4} and hence it cannot be extended to a

35-search strategy for T. Second, the number of searchers used for guarding in a particular move of $\mathcal{S}(T_{u_i})$, $i = 1, \ldots, 12$, is context dependent. For example $\mathcal{S}(T_{u_1})[1]$ in a search strategy $\langle r, u_1\rangle \oplus \langle r, u_3\rangle \oplus \mathcal{S}(T_{u_1})$ would use 36 searchers for guarding, because $w(r) + w(u_1) + w(u_3) = 36$.

Note that the number of searchers distinguished for each move denotes the minimum number of searchers, e.g. the label that represents $\mathcal{S}[1]$ that clears ru_1 equals $12 + 19$, even if $\mathcal{S}$ uses 35 searchers in total. The remaining 4 searchers, although occupying r, are not needed to perform $\mathcal{S}[1]$. In the remaining parts of this paper, similarly as in this example, we without loss of generality refer with $|\mathcal{S}[i]|$, where $\mathcal{S}$ is a search strategy, to the minimum number of searchers that is sufficient to perform $\mathcal{S}[i]$. This in particular implies that the value of k in Eq. (1) can be replaced with $w(h)$, where h is the homebase of $\mathcal{S}$.

3.1. Greedy search strategies

In this section we introduce the main tool in our analysis, i.e. the concept of a greedy search strategy, and then we list two facts that give the basic properties of greedy search strategies.

Definition 1. Let T be a weighted rooted tree. We say that a partial connected search strategy $\mathcal{S}$ for T is *greedy* if

(i) $\mathcal{S} \notin \varepsilon\langle T\rangle$, and
(ii) $w(u) \geq w(\delta(\mathcal{S}) \cap V(T_u))$ for each $u \in C_V(\mathcal{S})$, and
(iii) $E_u \cap C_E(\mathcal{S}) = \emptyset$ or $E_u \subseteq C_E(\mathcal{S})$ for each $u \in C_V(\mathcal{S})$.

Note that we in particular have that for a greedy search strategy $\mathcal{S}$ it holds $w(\delta(\mathcal{S})) \leq w(r)$, where r is the root of T. It also follows from the definition that a greedy search strategy is partial (and hence connected) and its homebase is the root of T. Moreover, each search strategy that clears all edges of T is greedy, because its border is empty. However, there may exist a greedy search strategy that does not clear all edges of T, but uses fewer searchers than each connected search strategy for T.

Note that the search strategy given in Fig. 1 satisfies conditions (i) and (iii), but it is not greedy, because $w(r) < w(\delta(\mathcal{S}))$ (note that $u = r$ is the only vertex in $C_V(\mathcal{S})$ for which condition (ii) fails). However, $\mathcal{S}(T_{u_i})$ is greedy for T_{u_i} for each $i = 1, \ldots, 12$.

Lemma 2. *Let T be a weighted rooted tree and let k be an integer. If $\mathcal{S}$ is a partial connected search strategy for T that can be k-extended to a greedy search strategy $\mathcal{S}'$ for T, and $\mathcal{S}_v$ is a greedy search strategy for a connected component T'_v in $T - C_E(\mathcal{S})$ rooted at v, where $v \in \delta(\mathcal{S})$, then $\mathcal{S} \oplus \mathcal{S}_v$ can be k-extended to a greedy search strategy $\widetilde{\mathcal{S}}$ for T such that*

(i) $C_E(\widetilde{\mathcal{S}}) = C_E(\mathcal{S}') \cup C_E(\mathcal{S}_v)$, *and*
(ii) $\widetilde{\mathcal{S}}$ *clears the edges in $C_E(\widetilde{\mathcal{S}}) \setminus C_E(\mathcal{S}_v)$ in the same order as $\mathcal{S}'$.*

Proof. Define $\widetilde{\mathcal{S}}$ so that $\widetilde{\mathcal{S}}[\preceq |\mathcal{S} \oplus \mathcal{S}_v|] = \mathcal{S} \oplus \mathcal{S}_v$ and the remaining moves of $\widetilde{\mathcal{S}}$ clear the edges in $C_E(\mathcal{S}') \setminus C_E(\mathcal{S} \oplus \mathcal{S}_v)$ in the same order as they are cleared by $\mathcal{S}'$. Clearly, $\widetilde{\mathcal{S}}$ is an extension of $\mathcal{S} \oplus \mathcal{S}_v$ that satisfies (i) and (ii). Now we argue that $\widetilde{\mathcal{S}}$ is a k-extension of $\mathcal{S} \oplus \mathcal{S}_v$. Suppose that $\mathcal{S}'[j']$ and $\widetilde{\mathcal{S}}[j]$ clear the same edge, where $j > |\mathcal{S} \oplus \mathcal{S}_v|$. Note that $j' > |\mathcal{S}|$. Let us analyze the sets $\delta(\mathcal{S}'[\preceq j']) \setminus \delta(\widetilde{\mathcal{S}}[\preceq j])$ and $\delta(\widetilde{\mathcal{S}}[\preceq j]) \setminus \delta(\mathcal{S}'[\preceq j'])$. Since $\widetilde{\mathcal{S}}[\preceq j]$ is 'mirroring' the moves of $\mathcal{S}'[\preceq j']$ on the edges not in $C_E(\mathcal{S}_v)$, we obtain that

$$\delta(\widetilde{\mathcal{S}}[\preceq \widetilde{j}]) \setminus \delta(\mathcal{S}'[\preceq j']) \subseteq \delta(\mathcal{S}_v),$$

and

$$\delta(\mathcal{S}'[\preceq j']) \setminus \delta(\widetilde{\mathcal{S}}[\preceq \widetilde{j}]) \subseteq C_V(\mathcal{S}_v) \setminus \delta(\mathcal{S}_v).$$

Moreover, each vertex in $\delta(\widetilde{\mathcal{S}}[\preceq \widetilde{j}]) \setminus \delta(\mathcal{S}'[\preceq j'])$ has an ancestor $u' \in \delta(\mathcal{S}'[\preceq j']) \setminus \delta(\widetilde{\mathcal{S}}[\preceq \widetilde{j}])$. Let U' be the set of all descendants of $u' \in \delta(\mathcal{S}'[\preceq j']) \setminus \delta(\widetilde{\mathcal{S}}[\preceq \widetilde{j}])$ such that $U' \subseteq \delta(\widetilde{\mathcal{S}}[\preceq \widetilde{j}]) \setminus \delta(\mathcal{S}'[\preceq j'])$. Since $\mathcal{S}_v$ is greedy, $w(U') \leq w(u')$. This proves that $w(\delta(\widetilde{\mathcal{S}}[\preceq \widetilde{j}])) \leq w(\delta(\mathcal{S}'[\preceq j'])) \leq k$. Consequently, by (1), $\mathtt{s}(\widetilde{\mathcal{S}}) \leq k$.

Finally we argue that $\widetilde{\mathcal{S}}$ is greedy. Let $u \in C_V(\widetilde{\mathcal{S}})$. If $u \notin C_V(\mathcal{S}_v)$, then, due to the fact that $\mathcal{S}'$ and $\mathcal{S}_v$ are greedy, $w(u) \geq w(\delta(\mathcal{S}') \cap V(T_u)) \geq w(\delta(\widetilde{\mathcal{S}}) \cap V(T_u))$. If $u \in C_V(\mathcal{S}_v)$, then $w(u) \geq w(\delta(\mathcal{S}_v) \cap V(T_u)) \geq w(\delta(\widetilde{\mathcal{S}}) \cap V(T_u))$, also because $\mathcal{S}'$ and $\mathcal{S}_v$ are greedy. □

Theorem 2. *Let T be a weighted rooted tree and let k be an integer. If $\mathcal{S}$ is a partial search strategy for T that can be k-extended to a greedy search strategy $\mathcal{S}'$ for T, $\mathcal{S} \neq \mathcal{S}'$, then*

(i) *there exists a greedy $(k - w(\delta(\mathcal{S}) \setminus \{v\}))$-search strategy $\mathcal{S}_v$ for T_v for some $v \in \delta(\mathcal{S})$ with $C_E(\mathcal{S}) \cap E_v = \emptyset$ such that $\mathcal{S} \oplus \mathcal{S}_v$ can be k-extended to a greedy search strategy $\widetilde{\mathcal{S}}$ for T, or*
(ii) *a search strategy $\mathcal{S} \oplus \langle v, u\rangle$, where $v \in \delta(\mathcal{S})$, $C_E(\mathcal{S}) \cap E_v \neq \emptyset$ and $uv \in E_v \setminus C_E(\mathcal{S})$, can be k-extended to a greedy search strategy $\widetilde{\mathcal{S}}$ for T, where*

both in (i) *and in* (ii) *for the search strategy $\widetilde{\mathcal{S}}$ it holds that $C_E(\widetilde{\mathcal{S}}) = C_E(\mathcal{S}')$, and the order of clearing the edges in E_u is the same both in $\widetilde{\mathcal{S}}$ and in $\mathcal{S}'$ for each $u \in C_V(\mathcal{S}') \setminus \delta(\mathcal{S}')$.*

Proof. Find the minimum integer $i \in \{|\mathcal{S}|+1, \ldots, |\mathcal{S}'|\}$ such that there exists $v \in \delta(\mathcal{S})$ that satisfies $(C_E(\mathcal{S}'[\preceq i]) \setminus C_E(\mathcal{S})) \cap E_v \neq \emptyset$ and

$$|E_u \cap C_E(\mathcal{S}'[\preceq i])| \in \{0, |E_u|\} \quad \text{and} \quad w(u) \geq w(\delta(\mathcal{S}'[\preceq i]) \cap V(T')) \quad \text{for each } u \in C_V(\mathcal{S}'[\preceq i]) \cap V(T'), \tag{2}$$

where T' is the connected component of $T - C_E(\mathcal{S})$ that contains the vertex v. Such an integer i exists, because $\mathcal{S}'$ is greedy and $\mathcal{S} \neq \mathcal{S}'$. Construct $\mathcal{S}_v$ by performing the clearing moves of the edges in $(C_E(\mathcal{S}'[\preceq i]) \setminus C_E(\mathcal{S})) \cap E(T')$ in the same order as in $\mathcal{S}'$. By the choice of v and i, $\mathcal{S}_v \notin \varepsilon\langle T'\rangle$.

First we prove that $|(\mathcal{S} \oplus \mathcal{S}_v)[j]| \leq k$ for each $j = |\mathcal{S}|+1, \ldots, |\mathcal{S}|+|\mathcal{S}_v|$. Let $j \in \{|\mathcal{S}|+1, \ldots, |\mathcal{S}|+|\mathcal{S}_v|\}$ be selected arbitrarily. Let j' be selected in such a way that $\mathcal{S}'[j']$ clears the same edge as $(\mathcal{S} \oplus \mathcal{S}_v)[j]$. Due to (1), it is enough to argue that $w(\delta((\mathcal{S} \oplus \mathcal{S}_v)[\preceq j])) \leq w(\delta(\mathcal{S}'[\preceq j']))$. Clearly, $\delta(\mathcal{S}'[\preceq j']) \cap V(T') = \delta((\mathcal{S} \oplus \mathcal{S}_v)[\preceq j]) \cap V(T')$. On the other hand,

$$w(\delta(\mathcal{S}'[\preceq j']) \setminus V(T')) \geq w(\delta((\mathcal{S} \oplus \mathcal{S}_v)[\preceq j]) \setminus V(T'))$$

due to the choice of i. Hence, $|(\mathcal{S} \oplus \mathcal{S}_v)[j]| \leq k$, and therefore by Lemma 1, $\mathcal{S}_v$ is a $(k - w(\delta(\mathcal{S}) \setminus \{v\}))$-search strategy for T'.

By (2) and by the fact that $\mathcal{S}_v \notin \varepsilon\langle T'\rangle$, $\mathcal{S}_v$ is greedy for T', which gives by Lemma 2 that if $C_E(\mathcal{S}) \cap E_v = \emptyset$, then (i) holds, and otherwise (ii) is satisfied. □

Informally speaking, if one wants to extend a partial search strategy $\mathcal{S}$ to a greedy one, then one of the two following extensions should be considered. The first extension is by clearing a contaminated edge $e \in E_v$, where E_v is selected so that some edges in E_v are already clear. The other extension is by finding a vertex $v \in \delta(\mathcal{S})$ such that none of the edges in E_v has been cleared (and thus no edge in T_v is clear) and by performing the moves of some greedy search strategy $\mathcal{S}_v$ for T_v. Note that Theorem 2 does not say how to find the edge e or the search strategy $\mathcal{S}_v$.

4. A generic algorithm for finding search strategies

Let T be a weighted rooted tree. Let $\pi : \{1, \ldots, |V_v|\} \to V_v$ be a permutation of the vertices in V_v for some $v \in V(T)$ such that $V_v \neq \emptyset$. We say that a search strategy $\mathcal{S}$ for T clears the edges in E_v *according to* π if $E_v \subseteq C_E(\mathcal{S})$ and $v\pi(i)$ is cleared prior to $v\pi(i+1)$ by $\mathcal{S}$ for each $i = 1, \ldots, |V_v| - 1$. Let L_T denote the set of vertices of degree 1 in T.

In this section we describe an algorithm that finds a connected search strategy for a given weighted tree. The search strategy is, in general, not optimal, because for each vertex we are given a set of permutations of the edges leading from the vertex to its children, and the final search strategy needs to respect this order, i.e. the edges have to be cleared according to one of the given permutations.

The algorithm is described in Section 4.1. Then, we argue in Section 4.2 that this generic algorithm finds a valid connected search strategy that uses the minimum number of searchers among all connected search strategies that follow the above restriction on the order of clearing the edges. Section 4.3 deals with the complexity, i.e. it is proven that the running time of the algorithm is polynomial in the size of the tree and in the size of the permutation set. Note that it follows in particular that the algorithm finds an optimal solution whenever all possible permutations are provided for each vertex, but then the running time is in general not polynomial in the size of the tree.

4.1. The generic algorithm

Before giving a formal description of the generic algorithm we will sketch its main ideas. The computation is performed by the main procedure called GWTAS (*Generic Weighted Tree Approximate Searching*) that finds a connected search strategy for the given tree T and for each possible homebase. Also, the input includes a function Γ that to each possible set V_v, $v \in V(T)$, assigns a set of permutations of the vertices in V_v. (Note that the set V_v depends on the choice of the root of T and Γ gives the permutation set for each v and for each choice of the root of T.) For a fixed homebase, the tree (rooted at the homebase) is processed by GWTAS in a bottom up manner. In particular, for each subtree T_v with $E_v \neq \emptyset$ the algorithm finds a partial search strategy denoted by $\mathcal{S}(T_v)$ by several invocations of a subroutine GSS (*Generic Subtree Searching*). When GSS is called for a subtree T_v, then it is guaranteed that $\mathcal{S}(T_u)$ has already been computed for each $u \in V(T_v) \setminus (\{v\} \cup L_T)$. We prove later that $\mathcal{S}(T_v)$ is a greedy search strategy for T_v such that there exists no greedy k-search strategy for $k < \mathtt{s}(\mathcal{S}(T_v))$ that clears the edges in each set E_u according to some permutation in Γ. Once $\mathcal{S}(T_v)$ is computed for each $v \in V(T)$ the algorithm finds a connected search strategy $\mathcal{S}$ for T (with the homebase being the root of T) by iteratively performing the following step. If $\mathcal{S}$ is the current partial search strategy, then some vertex $v \in \delta(\mathcal{S})$ is selected and the new current search strategy carried to the next iteration is $\mathcal{S} \oplus \mathcal{S}(T_v)$. The final search strategy is connected, the root of T is its homebase and the edges in E_v are cleared according to some permutation provided by Γ for each non-empty V_v.

First we describe the subroutine GSS (*Generic Subtree Searching*) used by the main procedure. The input is a tree T rooted at a vertex v, a permutation π of V_v and a set $\mathcal{A}$ that contains a partial search strategy $\mathcal{S}(T_u)$ for each $u \in V(T_v) \setminus (\{v\} \cup L_T)$. (The search strategies in $\mathcal{A}$ have been computed by previous calls to GSS.) The procedure GSS returns a partial search strategy $\mathcal{S}_\pi$ for T with homebase v that clears the edges in E_v according to the permutation π. Moreover, each subtree T_u, $u \in V_v$, is cleared by a concatenation of some search strategies in $\mathcal{A}$. As we prove later, $\mathcal{S}(T_v)$ is greedy and there exists no greedy search strategy for T_v that uses fewer than $\mathtt{s}(\mathcal{S}(T_v))$ searchers and satisfies the above conditions.

Before giving the pseudo-code of GSS we sketch its main steps, referring to the lines of the pseudo-code given below. Each iteration of the main loop of GSS tries to compute a partial search strategy $\mathcal{S}_\pi$ for T_v. The integer k_π is the number

of searchers available for $\mathcal{S}_\pi$. Initially all k_π searchers are placed at v (line 4) and the computation of $\mathcal{S}_\pi$ starts. The 'for' loop, that executes within the main loop, is responsible for computing $\mathcal{S}_\pi$. In the i-th iteration of this loop $w(\pi(i))$ searchers slide from v to $\pi(i)$ which results in clearing $u\pi(i)$ and $\pi(i)$ becomes guarded (line 7). If the latter is not possible, then the procedure decides that k_π searchers are not sufficient (line 6), k_π increases appropriately (line 13) and the new iteration of the main loop begins. Otherwise, the procedure repeatedly checks (in the 'while' loop in lines 8–10) whether there exists a search strategy $\mathcal{S}(T_u) \in \mathcal{A}$ for some $u \in \delta(\mathcal{S}_\pi) \setminus (\{v\} \cup L_T)$ such that $\mathcal{S}_\pi \oplus \mathcal{S}(T_u)$ does not use more then k_π searchers. The latter is equivalent to $\mathtt{s}(\mathcal{S}(T_u)) \leq k_\pi - w(\delta(\mathcal{S}_\pi) \setminus \{u\})$. If $\mathcal{S}(T_u)$ has been found, then $\mathcal{S}_\pi \oplus \mathcal{S}(T_u)$ becomes the new partial search strategy $\mathcal{S}_\pi$ (line 9). If $\mathcal{S}_\pi$ succeeds in clearing all edges in E_v, then the computation of $\mathcal{S}_\pi$ stops. Then, it is checked (line 14) whether $w(\delta(\mathcal{S}_\pi)) \leq w(v)$ and whether $\mathcal{S}_\pi$ clears all edges in E_v which (as we prove later) is equivalent to checking whether $\mathcal{S}_\pi$ is greedy. If the answer is affirmative, then the computation stops. Otherwise, k_π increases, and the computation of a new search strategy $\mathcal{S}_\pi$ begins for a larger value of k_π.

Note that $\mathcal{S}_\pi = \varepsilon\langle T_v, k_\pi\rangle$ satisfies $w(\delta(\mathcal{S}_\pi)) \leq w(v)$. Moreover, if $\mathcal{S}_\pi$ does not clear all edges in E_v, but has the property that if vu has been cleared, $u \in E_v$, then T_u is completely cleared by $\mathcal{S}_\pi$, then such an $\mathcal{S}_\pi$ also satisfies the latter inequality. However, in both cases $\mathcal{S}_\pi$ is not greedy and hence such search strategies are not accepted by GSS.

Procedure GSS(T, π, $\mathcal{A}$) (*Generic Subtree Searching*)
Input: a weighted tree $T = (V(T), E(T), w)$ rooted at v, where $|E(T)| > 0$, a permutation $\pi : \{1, \ldots, |V_v|\} \to V_v$ and a set $\mathcal{A}$ that contains $\mathcal{S}(T_u)$ for each $u \in V(T_v) \setminus (\{v\} \cup L_T)$.
Output: a partial search strategy for T_v.
begin
1: $k_\pi \leftarrow w(v)$.
2: **repeat**
3: Set $s_j \leftarrow +\infty$ for each $j = 1, \ldots, |E_v|$.
4: $\mathcal{S}_\pi \leftarrow \varepsilon\langle T_v, k_\pi\rangle$.
5: **for** $j \leftarrow 1$ **to** $|E_v|$ **do**
6: If $\mathtt{s}(\mathcal{S}_\pi \oplus \langle v, \pi(j)\rangle) > k_\pi$, then let $s_j \leftarrow \mathtt{s}(\mathcal{S}_\pi \oplus \langle v, \pi(j)\rangle)$, $x_j \leftarrow v$, and **goto** line 11.
7: $\mathcal{S}_\pi \leftarrow \mathcal{S}_\pi \oplus \langle v, \pi(j)\rangle$.
8: **while** $\exists u \in \delta(\mathcal{S}_\pi) \setminus \{v\}$ such that $\mathtt{s}(\mathcal{S}(T_u)) \leq k_\pi - w(\delta(\mathcal{S}_\pi) \setminus \{u\})$ **do**
9: $\mathcal{S}_\pi \leftarrow \mathcal{S}_\pi \oplus \mathcal{S}(T_u)$.
10: **end while**
11: If $\delta(\mathcal{S}_\pi) \setminus \{v\} \neq \emptyset$, then find $u \in \delta(\mathcal{S}_\pi) \setminus \{v\}$ with the minimum $k' = w(\delta(\mathcal{S}_\pi) \setminus \{u\}) + \mathtt{s}(\mathcal{S}(T_u))$ and set $s_j \leftarrow k'$, $x_j \leftarrow u$.
12: **end for**
13: If $\delta(\mathcal{S}_\pi) \neq \emptyset$, then find $i \in \{1, \ldots, |E_v|\}$ such that $x_i \in \delta(\mathcal{S}_\pi)$ and $s_i \leq s_j$ for each $j = 1, \ldots, |E_v|$ such that $x_j \in \delta(\mathcal{S}_\pi)$, and set $k_\pi \leftarrow s_i$.
14: **until** $w(\delta(\mathcal{S}_\pi)) \leq w(v)$ and $\mathcal{S}_\pi \notin \varepsilon\langle T_v\rangle$.
15: **return** $\mathcal{S}_\pi$.
end procedure GSS.

Let T be a weighted tree. Define

$$\mathcal{V}_v = \{N_T(v) \setminus \{u\} \colon |N_T(v)| > 1 \text{ and } u \in N_T(v)\} \cup \{N_T(v)\}$$

and let $\mathcal{V} = \bigcup_{v \in V(T)} \mathcal{V}_v$. Informally speaking, $\mathcal{V}$ is constructed so that for each choice of the roof for T and for each $v \in V(T)$ with non-empty V_v it holds that $V_v \in \mathcal{V}$. We say that Γ is an *ordering* for T if Γ assigns to each set $X \in \mathcal{V}$ a non-empty set $\Gamma(X)$ of permutations of X.

The input to the procedure GWTAS is an unrooted tree T and an ordering Γ for T. (If a permutation $\pi \in \Gamma(V_v)$ has been selected while computing a search strategy, then the edges in E_v are cleared according to π by the search strategy.) The set $\mathcal{X}'$, initially set to be empty (line 1), is extended during the execution of GWTAS with the best connected search strategy found for each choice of the root (line 21).

Procedure GWTAS(T, Γ) (*Generic Weighted Tree Approximate Searching*)
Input: a weighted tree T and an ordering Γ for T.
Output: a connected search strategy for T.
begin
1: $\mathcal{X}' \leftarrow \emptyset$.
2: **for each** $r \in V(T)$ **do**

Consider T to be rooted at r.
Let $(v_1, \ldots, v_n)$ be any post-ordering (each child precedes its parent) of the vertices of T.
$\mathcal{A} \leftarrow \emptyset$.
for $i \leftarrow 1$ **to** n **do**
  if $V_{v_i} \neq \emptyset$ **then**
    $\mathcal{X} \leftarrow \emptyset$.
    for each $\pi \in \Gamma(V_{v_i})$ **do**
      Add to $\mathcal{X}$ the search strategy returned by $\texttt{GSS}(T_{v_i}, \pi, \mathcal{A})$.
    end for
    $\mathcal{S}(T_{v_i}) \leftarrow \mathcal{S}$, where $\mathcal{S} \in \mathcal{X}$ and $\mathtt{s}(\mathcal{S}) = \min\{\mathtt{s}(\mathcal{S}') : \mathcal{S}' \in \mathcal{X}\}$.
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathcal{S}(T_{v_i})\}$.
  end if
end for
Let $\mathcal{S}_r \leftarrow \mathcal{S}(T_r)$.
while $C_E(\mathcal{S}_r) \neq E(T)$ **do**
  Find $v \in \delta(\mathcal{S}_r)$ with the minimum $w(\delta(\mathcal{S}_r) \setminus \{v\}) + \mathtt{s}(\mathcal{S}(T_v))$.
  $\mathcal{S}_r \leftarrow \mathcal{S}_r \oplus \mathcal{S}(T_v)$.
end while
$\mathcal{X}' \leftarrow \mathcal{X}' \cup \{\mathcal{S}_r\}$.
end for
return $\mathcal{S}$ such that $\mathcal{S} \in \mathcal{X}'$ and $\mathtt{s}(\mathcal{S}) = \min\{\mathtt{s}(\mathcal{S}') : \mathcal{S}' \in \mathcal{X}'\}$.
end procedure GWTAS.

In order to simplify the presentation, we skipped some possible optimizations in the pseudo-codes of GSS and GWTAS. Those details are included in the proofs of Lemmas 7 and 8, where we analyze the complexities of both procedures.

4.2. The correctness of the generic algorithm

Let Γ be an ordering for a weighted tree T. If $\mathcal{S}_v$ is a partial search strategy for T_v that clears the edges in E_u according a permutation in $\Gamma(V_u)$ for each $u \in C_V(\mathcal{S}_v) \setminus \delta(\mathcal{S}_v)$, then we say that $\mathcal{S}_v$ is a Γ-strategy. We say that a set $\mathcal{A}$ is *Γ-complete* for T_v if for each $u \in V(T_v) \setminus (\{v\} \cup L_T)$ the two following conditions hold:

- a greedy search Γ-strategy $\mathcal{S}(T_u)$ belongs to $\mathcal{A}$, and
- if $\mathcal{S}_u$ is any greedy search Γ-strategy for T_u, then $\mathtt{s}(\mathcal{S}(T_u)) \leq \mathtt{s}(\mathcal{S}_u)$.

In other words, if $\mathcal{A}$ is Γ-complete for T, then it contains a greedy search Γ-strategy for each subtree T_u, $u \notin L_T$, where u is a descendant of v in T_v, and there exists no greedy search Γ-strategy for T_u that uses fewer searchers. If we in the following say that Γ is an ordering for a rooted tree, then we point out that $\Gamma(V_v)$ refers to the set V_v with respect to the given choice of the root.

We start with a lemma that gives a necessary and sufficient condition for any search strategy constructed as in the procedure GSS to be greedy. The condition requires to compare the weight of the root of the subtree with the weight of the border of the search strategy. In other words, it is enough to check condition (ii) in Definition 1 only for u being the root of T.

Lemma 3. *Let T be a tree rooted at v and let $\mathcal{S} = \mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_p$, $p > 0$, be a search strategy for T such that $E_v \subseteq C_E(\mathcal{S})$, $\mathcal{S}_1 = \langle v, u\rangle$ for some $u \in V_v$ and for each $i = 2, \ldots, p$ we have that $\mathcal{S}_i = \langle v, u\rangle$ for some $u \in V_v$ or $\mathcal{S}_i$ is a greedy search strategy for T_u for some $u \in \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{i-1}) \setminus \{v\}$. Then, $\mathcal{S}$ is greedy if and only if $w(v) \geq w(\delta(\mathcal{S}))$.*

Proof. Note that the 'only if' part is trivial, so we prove the 'if part'.

Since $p > 0$, it holds $\mathcal{S} \notin \varepsilon\langle T_v\rangle$. By the construction of $\mathcal{S}$, it also satisfies (iii) in Definition 1. Let $u \in C_V(\mathcal{S})$ be selected arbitrarily. We have to prove that $w(u) \geq w(\delta(\mathcal{S}) \cap V(T_u))$. This inequality trivially follows for each $u \in L_{T_v} \setminus \{v\}$. Moreover, it follows for $u = v$ by assumption. Thus, let $u \notin \{v\} \cup L_{T_v}$. Let $\mathcal{S}_{\pi,u}$ be the search strategy $\mathcal{S}$ restricted to the subtree T_u. Since $u \notin L_{T_v}$, $\mathcal{S}_{\pi,u}$ clears a non-empty subtree of T_u and we without loss of generality have that $\mathcal{S}_{\pi,u} = \mathcal{S}_{p_1} \oplus \cdots \oplus \mathcal{S}_{p_l}$, where $\mathcal{S}_{p_i}$ is a greedy search strategy for T_{u_i}, $u_i \in V(T_u)$.

We prove by induction on $i = 1, \ldots, l$ that $\mathcal{S}_{p_1} \oplus \cdots \oplus \mathcal{S}_{p_i}$ is greedy. By assumption, $\mathcal{S}_{p_1}$ is greedy, so let us assume that the induction hypothesis holds for some i, $1 \leq i < l$, and we prove the claim for $i + 1$. Let $x \in C_V(\mathcal{S}_{p_1} \oplus \cdots \oplus \mathcal{S}_{p_{i+1}})$ be selected arbitrarily. We consider two following cases.

Case 1: $V(T_x) \cap \delta(\mathcal{S}_{p_{i+1}}) = \emptyset$. Then, by the fact that $\mathcal{S}_{p_1} \oplus \cdots \oplus \mathcal{S}_{p_i}$ is greedy we obtain

$$w(x) \geq w(\delta(\mathcal{S}_{p_1} \oplus \cdots \oplus \mathcal{S}_i) \cap V(T_x)) = w(\delta(\mathcal{S}_{p_1} \oplus \cdots \oplus \mathcal{S}_{p_{i+1}}) \cap V(T_x)).$$

Case 2: $V(T_x)\cap\delta(\mathcal{S}_{p_{i+1}})\neq\emptyset$. If $x\in C_V(\mathcal{S}_{p_{i+1}})$, then by the fact that $\mathcal{S}_{p_{i+1}}$ is greedy we obtain that $w(x)\geq w(\delta(\mathcal{S}_{p_{i+1}})\cap V(T_x))$. Moreover, $\delta(\mathcal{S}_{p_1}\oplus\cdots\oplus\mathcal{S}_{p_{i+1}})\cap V(T_x)=\delta(\mathcal{S}_{p_{i+1}})\cap V(T_x)$ and in such case the claim follows. If, on the other hand, $x\notin C_V(\mathcal{S}_{p_{i+1}})$, then by the induction hypothesis, $w(x)\geq w(\delta(\mathcal{S}_{p_1}\oplus\cdots\oplus\mathcal{S}_{p_i})\cap V(T_x))$. Since x is an ancestor of u_{i+1}, $u_{i+1}\in\delta(\mathcal{S}_{p_1}\oplus\cdots\oplus\mathcal{S}_{p_i})$, and $w(u_{i+1})\geq w(\delta(\mathcal{S}_{p_{i+1}})\cap V(T_{u_{i+1}}))$ (that holds because $\mathcal{S}_{p_{i+1}}$ is greedy), we obtain

$$\begin{aligned}w(x)&\geq w(\delta((\mathcal{S}_{p_1}\oplus\cdots\oplus\mathcal{S}_{p_i})\setminus\{u_{i+1}\})\cap V(T_x))+w(u_{i+1})\\&\geq w(\delta((\mathcal{S}_{p_1}\oplus\cdots\oplus\mathcal{S}_{p_i})\setminus\{u_{i+1}\})\cap V(T_x))+w(\delta(\mathcal{S}_{p_{i+1}})\cap V(T_{u_{i+1}}))\\&=w(\delta((\mathcal{S}_{p_1}\oplus\cdots\oplus\mathcal{S}_{p_i})\setminus\{u_{i+1}\})\cap V(T_x))+w(\delta(\mathcal{S}_{p_{i+1}})\cap V(T_x))\\&=w(\delta(\mathcal{S}_{p_1}\oplus\cdots\oplus\mathcal{S}_{p_{i+1}})\cap V(T_x)).\end{aligned}$$

The latter holds, because due to condition (iii) in Definition 1, $u_{i+1}\notin\delta(\mathcal{S}_{p_1}\oplus\cdots\oplus\mathcal{S}_{p_{i+1}})$.

By the above, $\mathcal{S}_{\pi,u}$ is greedy, which in particular gives that $w(u)\geq w(\delta(\mathcal{S}_{\pi,u}))$. Since $\delta(\mathcal{S}_{\pi,u})=\delta(\mathcal{S})\cap V(T_u)$, we obtain that $w(u)\geq w(\delta(\mathcal{S})\cap V(T_u))$ as desired. □

Lemma 4 given below states that the execution of GSS always stops, and that GSS, for the given T_v, π and $\mathcal{A}$, returns a greedy search strategy $\mathcal{S}_\pi$ that clears the edges in E_v according to π, and the edges in E_u are cleared by $\mathcal{S}_\pi$ according to an ordering used by one of the search strategies in $\mathcal{A}$. Then, in Lemma 5, we prove that $\mathcal{S}_\pi$ uses the minimum number of searchers.

Lemma 4. *Let Γ be an ordering for a tree T rooted at v, where $\Gamma(V_v)=\{\pi\}$ for some permutation π. If $\mathcal{A}$ is Γ-complete for T, then* GSS*, for the given T, π and $\mathcal{A}$, returns a greedy search Γ-strategy for T.*

Proof. First observe that the computation of GSS stops and that the procedure returns a search strategy $\mathcal{S}_\pi$. Indeed, if the computation of $\mathcal{S}_\pi$ fails in a given iteration of the main loop, then k_π increases (line 13) and the new iteration begins. For $k_\pi\geq w(V(T))$, the procedure GSS is able to find a search strategy that clears all edges of T_v (regardless of the permutation π used) and therefore its border is an empty set which implies that such a search strategy is greedy. This guarantees by Lemma 3 that the condition of the 'repeat-until' loop is met (line 14) and the main loop finishes its execution.

The fact that $\mathcal{S}_\pi$ clears the edges in E_v according to π is obvious (see the 'for' loop in lines 5–12). Since $\mathcal{A}$ is Γ-complete for T we obtain that $\mathcal{S}_\pi$ is a Γ-strategy. Finally note that $\mathcal{S}_\pi\notin\varepsilon\langle T_v\rangle$ and Lemma 3 give the thesis. □

Lemma 5. *Let T be a weighted tree rooted at v. Let Γ be an ordering for the rooted tree T with $\Gamma(V_v)=\{\pi\}$. If $\mathcal{A}$ is Γ-complete for T_v and there exists a greedy k-search Γ-strategy for T_v, then* GSS *returns, for the given T, π and $\mathcal{A}$, a greedy search Γ-strategy $\mathcal{S}_\pi$ such that $\mathtt{s}(\mathcal{S}_\pi)\leq k$.*

Proof. By construction the search strategy $\mathcal{S}_\pi$ computed by GSS is of the form

$$\mathcal{S}_\pi=\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_l,$$

where $\mathcal{S}_i$ either equals $\mathcal{S}(T_u)$ for some $u\in\delta(\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_{i-1})$, $u\neq v$, or $\mathcal{S}_i=\langle v,u\rangle$ for some $u\in V_v$ (see lines 7 and 9 of GSS that contain the instructions that extend $\mathcal{S}_\pi$). By Lemma 4, $\mathcal{S}_\pi$ is a greedy search Γ-strategy.

Let $\mathcal{S}_v$ be a greedy search Γ-strategy for T_v. Assume that $k_\pi=\mathtt{s}(\mathcal{S}_v)$. We prove by induction that for each $i=1,\ldots,l$, $\mathtt{s}(\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_i)\leq k_\pi$ and the search strategy $\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_i$ can be k_π-extended to a greedy search Γ-strategy for T_v. The claim clearly follows for $i=1$, because $\mathcal{S}_1$ clears $v\pi(1)$ and so does $\mathcal{S}_v$.

Let now $i\in\{1,\ldots,l-1\}$ and we prove the claim for $i+1$. If $\mathcal{S}_{i+1}$ is a search strategy for T_u, $u\in V(T)\setminus\{v\}$, then by assumption $\mathcal{S}_{i+1}$ is a greedy search Γ-strategy. By the formulation of GSS (line 8), $\mathtt{s}(\mathcal{S}_{i+1})\leq k_\pi-w(\delta(\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_i)\setminus\{u\})$. Hence, by Lemma 1 and by the induction hypothesis, $\mathtt{s}(\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_{i+1})\leq k_\pi$. By Lemma 2 and by the induction hypothesis, $\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_{i+1}$ can be k_π-extended to a greedy search Γ-strategy.

If $\mathcal{S}_{i+1}$ clears an edge in E_v, then by the fact that $\mathcal{A}$ is Γ-complete for T_v we obtain by Theorem 2 that for each $u\in\delta(\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_i)$, such that $C_E(\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_i)\cap E_u=\emptyset$, there exists no greedy $(k_\pi-w(\delta(\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_i)\setminus\{u\}))$-search Γ-strategy for T_u. The latter is checked in line 8 of GSS. Thus, $\mathcal{S}_{i+1}$ has to clear an edge in E_u such that $u\in\delta(\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_i)$ and $C_E(\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_i)\cap E_u\neq\emptyset$. However, $u=v$ is the only vertex that satisfies the latter condition. By Theorem 2(ii), $\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_{i+1}$ can be k_π-extended to a greedy search strategy for T_v, and by the formulation of GSS, $\mathtt{s}(\mathcal{S}_1\oplus\cdots\oplus\mathcal{S}_{i+1})\leq k_\pi$ (lines 6 and 7).

It remains to argue that GSS sets the value of k_π to $\mathtt{s}(\mathcal{S}_v)$, where $\mathcal{S}_v$ is a greedy Γ-strategy such that there exists no greedy $(\mathtt{s}(\mathcal{S}_v)-1)$-search Γ-strategy for T_v. Note that if $\mathtt{s}(\mathcal{S}_v)=w(v)$, then the claim follows (due to the initialization in line 1), so assume that $\mathtt{s}(\mathcal{S}_v)>w(v)$, which in particular implies that the first iteration of the main loop of GSS does not find the desired search strategy $\mathcal{S}_\pi$. Let us consider any iteration of the main loop of GSS and let k'_π be the value of k_π in this particular iteration, while $\mathcal{S}'=\mathcal{S}_\pi$ is the search strategy obtained at the end of the iteration. If, for the corresponding integer $k_\pi=k'_\pi$, GSS does not find a desired search strategy $\mathcal{S}_\pi$, then by the formulation of GSS the next value of k_π, denoted by k', equals s_i for some $i\in\{1,\ldots,|E_v|\}$ (line 11). By the choice of k' (lines 6,11 and 13), the result of the execution of the main loop in lines 3–14 is identical for each $k_\pi=k'_\pi,\ldots,k'-1$. Moreover, if $k_\pi=k'$, then during some iteration of the nested 'for' loop in lines 5–12 either an edge $e\in E_v$ becomes clear such that $e\notin C_E(\mathcal{S}')$ (line 7), or a search strategy $\mathcal{S}(T_u)\in\mathcal{A}$ is used, $u\in\delta(\mathcal{S}')$ (line 9). In both cases, the final search strategy obtained at the end of this iteration of the main loop with $k_\pi=k'$ clears more edges than $\mathcal{S}'$. Thus, after a finite number of steps the variable k_π is set to the minimum value that guarantees that a greedy search Γ-strategy is computed by GSS. □

In order to simplify the statements, we extend the notion of the connected search number for Γ-strategies. In particular, given a weighted tree T, a vertex r of T and an ordering Γ for T, the symbol $\mathtt{cs}(T, r, \Gamma)$ denotes the minimum integer k such that there exists a connected k-search Γ-strategy for T with homebase r. Then, $\mathtt{cs}(T, \Gamma) = \min\{\mathtt{cs}(T, r, \Gamma) : r \in V(T)\}$.

Lemma 6. *If T is a weighted tree and Γ is an ordering for T, then at the end of the execution of* GWTAS *for T and Γ it holds*

(i) *for each choice of the root for T and for each subtree T_v, $|V(T_v)| > 1$, a greedy search Γ-strategy $\mathcal{S}_v$ has been computed, and $\mathcal{S}_v$ uses the minimum number of searchers among all greedy search Γ-strategies for T_v,*
(ii) GWTAS *returns a connected $\mathtt{cs}(T, \Gamma)$-search Γ-strategy for T.*

Proof. We continue with the assumption that T is rooted at an arbitrary vertex r. In the following we refer to the lines of the pseudo-code of GWTAS.

We prove (i) by induction on the subtree height. Let for brevity $v = v_i, i \in \{1, \ldots, n\}$. If T_v satisfies $E(T_v) = E_v$ (i.e. T_v is of height 1), then the claim follows, because GWTAS tries to clear the edges in E_v according to all permutations in $\Gamma(V_v)$ by making the appropriate calls to GSS (lines 9–11). Thus, assume that T_v is of height greater than 1. By the induction hypothesis and by the formulation of GWTAS (the vertices of T are processed in post-order), we obtain that for each $u \in V_v$ the subtree T_u satisfies (i), i.e. if $|V(T_u)| > 1$, then $\mathcal{S}_u$ equals $\mathcal{S}(T_u)$ that belongs to $\mathcal{A}$ when GSS is called for T_v. We will analyze the computation of $\mathcal{S}(T_v)$.

The procedure GSS is called for T_v, for each permutation $\pi \in \Gamma(V_v)$ and a collection $\mathcal{A}$ that is Γ-complete for T_v (lines 9–11). The latter follows from the order of processing of the vertices of T by GWTAS (fixed in line 4). By Lemma 5 we obtain that the search strategy $\mathcal{S}_\pi$ returned by GSS is a greedy search Γ-strategy that uses the minimum number of searchers and clears the edges in E_v according to π. The thesis follows from the observations that $\mathcal{S}(T_v)$ is selected to be a strategy using the minimum number of searchers among all search strategies $\mathcal{S}_\pi$ returned by GSS (line 12), and GSS is called for each $\pi \in \Gamma(V_v)$. Then, $\mathcal{S}(T_v)$ is added to $\mathcal{A}$ (line 13).

Now we prove (ii). To that end it is enough to argue that $\mathcal{S}_r$, constructed for an arbitrary choice of r, is a connected search Γ-strategy for the rooted tree T with homebase r and $\mathtt{s}(\mathcal{S}_r) \leq \mathtt{cs}(T, r, \Gamma)$. Then, the thesis follows from the fact that $\mathcal{S}_r$ is added to $\mathcal{X}'$ (line 21), and from the choice of $\mathcal{S}$ returned by GWTAS (line 23). To prove the inequality we show by induction on the number of iterations of the 'while' loop (lines 17–20) of GWTAS responsible for the computation of $\mathcal{S}_r$ that the partial search strategy, denoted by $\mathcal{S}_r^i$, obtained at the end of the i-th iteration, can be $\mathtt{cs}(T, r, \Gamma)$-extended to a connected search Γ-strategy for T with homebase r and that $\mathcal{S}_r^i \neq \mathcal{S}_r^{i-1}$. We take $\mathcal{S}_r^0 = \mathcal{S}(T_r)$. Note that the cases of $i = 0$ and $i > 0$ are analogous, so we consider the latter one. (For $i = 0$ it is enough to take $\mathcal{S}_r^{-1} = \varepsilon\langle T, \mathtt{cs}(T, r, \Gamma)\rangle$ that clearly can be extended to a connected $\mathtt{cs}(T, r, \Gamma)$-search strategy that clears T.)

Suppose that the claim holds for $\mathcal{S}_r^i$, $i \geq 0$, and we consider the $(i+1)$-st iteration. By the induction hypothesis, $\mathtt{s}(\mathcal{S}_r^i) \leq \mathtt{cs}(T, r, \Gamma)$ and $\mathcal{S}_r^i$ can be $\mathtt{cs}(T, r, \Gamma)$-extended to a connected search strategy $\mathcal{S}'$ that clears T. Note that $\mathcal{S}'$ is greedy, because $\delta(\mathcal{S}') = \emptyset$. Since $C_E(\mathcal{S}_r^i) \cap E_v = \emptyset$ for each $v \in \delta(\mathcal{S}_r^i)$, by Theorem 2 we obtain that there exist $v \in \delta(\mathcal{S}_r^i))$ and a greedy $(\mathtt{cs}(T, r, \Gamma) - w(\delta(\mathcal{S}_r^i) \setminus \{v\}))$-search strategy $\mathcal{S}$ such that $\mathcal{S}_r^i \oplus \mathcal{S}$ can be $\mathtt{cs}(T, r, \Gamma)$-extended to a connected search strategy that clears T. By (i) we have that a greedy search Γ-strategy $\mathcal{S}_v$ for T_v has been computed by GWTAS, i.e. $\mathcal{S}_v \in \mathcal{A}$, and $\mathtt{s}(\mathcal{S}_v) \leq \mathtt{s}(\mathcal{S})$. Therefore, by Lemma 1 and by the induction hypothesis,

$$\mathtt{s}(\mathcal{S}_r^i \oplus \mathcal{S}_v) = \max\{\mathtt{s}(\mathcal{S}_r^i), w(\delta(\mathcal{S}_r^i) \setminus \{v\}) + \mathtt{s}(\mathcal{S}_v)\} \leq \max\{\mathtt{cs}(T, r, \Gamma), \mathtt{s}(\mathcal{S}_r^i \oplus \mathcal{S})\} \leq \mathtt{cs}(T, r, \Gamma).$$

By Lemma 2, $\mathcal{S}_r^i \oplus \mathcal{S}_v$ can be $\mathtt{cs}(T, r)$-extended to a connected search strategy that clears T. □

We finish this section with a theorem that follows from the procedures GSS and GWTAS. The theorem provides a structural characterization of greedy search strategies and it will be useful in the last section where we give the 3-approximation algorithm.

Theorem 3. *Let T be a weighted tree rooted at r, let Γ be an ordering for T and let $d = |E_r|$. If there exists a greedy k-search Γ-strategy for T that clears the edges in E_r according to a permutation $\pi \in \Gamma(V_r)$, then there exists a greedy k-search Γ-strategy $\mathcal{S}$ that clears the same subgraph and satisfies*

$$\mathcal{S} = \mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_l,$$

where $l \geq d$ and

$$\mathcal{S}_i = \mathcal{S}_i^1 \oplus \cdots \oplus \mathcal{S}_i^{t_i}, \quad i = 1, \ldots, d,$$

where $t_i \geq 1$, $\mathcal{S}_i^1 = \langle r, \pi(i)\rangle$ and $\mathcal{S}_i^j$, $j = 2, \ldots, t_i$, is a greedy search Γ-strategy for T_v for some $v \in V(T_{\pi(i)})$ for each $i = 1, \ldots, d$. Moreover, $\mathcal{S}_i$ is a greedy search Γ-strategy for T_v for some $v \in \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{i-1})$ for each $i = d+1, \ldots, l$. □

In other words, $\mathcal{S}_i$, $i \leq d$, clears $r\pi(i)$ and a subtree of $T_{\pi(i)}$. Moreover, for each $i = d+1, \ldots, l$ the search strategy $\mathcal{S}_i$ for T_v is greedy, where $v \in \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{i-1})$. Note that we may without loss of generality assume that $t_d = 1$, or in other words, that $\mathcal{S}_d$ clears only the edge $r\pi(d)$ and then the search strategies $\mathcal{S}_{d+1}, \ldots, \mathcal{S}_l$ follow. In section Section 5 we will work with this assumption in order to simplify the statements.

4.3. The complexity of the generic algorithm

In this section we analyze the running time of GWTAS and we start by proving that GSS runs in $O(n \log n)$-time for an n-node tree.

Lemma 7. *The running time of* GSS *is $O(n \log n)$ for the given input rooted tree T, permutation π and a Γ-complete set $\mathcal{A}$ for T, where $n = |V(T)|$.*

Proof. First we analyze the running time of the 'for' loop (lines 5–12). For each subtree T_u the search strategy $\mathcal{S}(T_u)$ can be accessed in constant time. Also, the vertices in $\delta(\mathcal{S}_\pi)$ can be stored in a min-heap H, where the key value associated with a vertex $u \in \delta(\mathcal{S}_\pi)$ is $\mathtt{s}(\mathcal{S}(T_u)) - w(u)$. Then, finding the search strategy $\mathcal{S}(T_u)$ in line 8 requires $O(1)$ time, because the condition $\mathtt{s}(\mathcal{S}(T_u)) \leq k_\pi - w(\delta(\mathcal{S}_\pi) \setminus \{u\})$ is equivalent to $\mathtt{s}(\mathcal{S}(T_u)) - w(u) \leq k_\pi - w(\delta(\mathcal{S}_\pi))$, because $w(\delta(\mathcal{S}_\pi) \setminus \{u\}) = w(\delta(\mathcal{S}_\pi)) - w(u)$. Since the right hand side of this inequality does not depend on u, the minimal element of H determines a search strategy in $\mathcal{A}$ that can be used in each iteration of the 'while' loop. The computation of $\mathcal{S}_\pi \oplus \mathcal{S}(T_u)$ (line 9) can be performed in time $O(|\mathcal{S}(T_u)|)$. If the choice of k_π does not allow to clear an edge $v\pi(j)$ for some $j \in \{1, \ldots, |E_v|\}$, then it is not necessary to recalculate in any iteration of the 'for' loop the part of the search strategy computed in the previous iteration. Indeed, for each $j = 1, \ldots, |E_v|$ the j-th iteration may start by using the part of the previous search strategy that clears all edges 'between' clearing $v\pi(j)$ and $v\pi(j+1)$. After each extension of $\mathcal{S}_\pi$ with $\mathcal{S}(T_u)$ we add to the heap H each vertex in $\delta(\mathcal{S}(T_u))$ and we remove u from H.

The total number of search strategies in $\mathcal{A}$ used to construct the final search strategy $\mathcal{S}_\pi$ is $O(n)$, which implies that the total time required for insertions to and removals form H is $O(n \log n)$. The total time to construct $\mathcal{S}_\pi$ itself is linear, i.e. $O(n)$, because the total number of clearing moves of any partial (monotone) search strategy for T is at most $|E(T)| < n$. This completes the proof. □

The running time of GWTAS clearly depends on the number of permutations in the given ordering Γ for T. We introduce the value of γ that is assumed to bound the size of each permutation set, i.e. $|\Gamma(X)| \leq \gamma$ for each $X \in \mathcal{V}$.

Lemma 8. *Let T and Γ be an input to* GWTAS*. The running time of* GWTAS *is $O(\gamma n^3 \log n)$, where $n = |V(T)|$ and $|\Gamma(X)| \leq \gamma$ for each $X \in \mathcal{V}$.*

Proof. The time required to compute $\mathcal{S}(T_v)$ that is added to $\mathcal{A}$ (lines 7–14) is $O(|\Gamma(V_v)|p)$, where p is the complexity of GSS, because for this purpose the procedure GSS is called for each permutation in $\Gamma(V_v)$. Thus, by Lemma 7, the time complexity of finding $\mathcal{S}(T_v)$ is $O(|\Gamma(V_v)|n \log n) = O(\gamma n \log n)$. Also, at the beginning of each iteration of the nested 'for' loop of GWTAS (lines 9–11) one can check whether $\mathcal{S}(T_{v_i})$ has been computed previously in order to avoid repeating the computations. The latter is possible, because the same subtree T_{v_i} may appear for different choices of the root. Note that there are $O(n^2)$ pairwise different non-empty sets V_v and subtrees T_v ($O(n)$ for each choice of the root of T). Thus, the computation of all search strategies $\mathcal{S}(T_v)$ requires $O(\gamma n^3 \log n)$ time. The computation of $\mathcal{S}_r$ for the given choice of the root r (lines 16–20) requires $O(n \log n)$ time and the analysis is analogous to the one for GSS. Thus, finding all connected search strategies $\mathcal{S}_r$ that clear T entirely and have different homebases $r \in V(T)$ requires $O(n^2 \log n)$ time in total and does not increase the overall complexity of GWTAS. □

5. An approximation algorithm

We say that a partial search strategy $\widetilde{\mathcal{S}}$ for a weighted tree T rooted at r is *simple* if $\widetilde{\mathcal{S}}$ is greedy and $\widetilde{\mathcal{S}} = \mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_d$, where $\mathcal{S}_i$ clears an edge $ru \in E_v$ and a subtree of T_u for each $i = 1, \ldots, d$ and $d = |E_r|$.

Informally speaking, a simple search strategy $\mathcal{S}$ for T is restricted in the following way. If a particular move of $\mathcal{S}$ clears an edge ru, $u \in V_r$, then the following moves clear some edges of T_u. However, once the next edge in E_r becomes clear, the strategy is not allowed to clear any edges of T_u in its future moves. There exist trees for which each simple search strategy uses more searchers than k searchers, where k is the number of searchers used by a greedy search strategy that uses the minimum number of searchers (see e.g. the tree in Fig. 1).

Let for brevity $d = |E_r|$. Let $\mathcal{S}$ be a greedy search strategy. We may without loss of generality assume that $\mathcal{S}$ has the form as in Theorem 3. Recall that $\mathcal{S}_i$, $i \leq d$, clears $r\pi(i)$ and a subtree of $T_{\pi(i)}$. Moreover, for each $i = d+1, \ldots, l$ the search strategy $\mathcal{S}_i$ for T_v is greedy, where $v \in \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{i-1})$. As argued before, we may assume without loss of generality that $t_d = 1$, i.e. $\mathcal{S}_d = \langle r, \pi(d)\rangle$. Note that the part of $\mathcal{S}$ composed with the first d search strategies, i.e. $\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_d$, is simple, but if $l > d$, then $\mathcal{S}$ is not simple in general.

Now we perform the following modification to $\mathcal{S}$ in order to convert it into a simple search strategy. Find the integer $i \in \{1, \ldots, d\}$ such that the greedy search strategy $\mathcal{S}_{d+1}$ clears a subtree of $T_{\pi(i)}$, i.e. $C_E(\mathcal{S}_{d+1}) \subseteq E(T_{\pi(i)})$. Then, perform the moves of $\mathcal{S}_{d+1}$ immediately prior to $\mathcal{S}_{i+1}$, i.e. let the new search strategy be as follows

$$\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_i \oplus \mathcal{S}_{d+1} \oplus \mathcal{S}_{i+1} \oplus \cdots \oplus \mathcal{S}_d \oplus \mathcal{S}_{d+2} \oplus \cdots \oplus \mathcal{S}_l.$$

Note that if $i = d$, then the above does not change $\mathcal{S}$. Repeat this step for the remaining search strategies $\mathcal{S}_{d+2}, \ldots, \mathcal{S}_l$. The final search strategy obtained in this way is denoted by

$$\widetilde{\mathcal{S}} = \widetilde{\mathcal{S}}_1 \oplus \cdots \oplus \widetilde{\mathcal{S}}_d, \tag{3}$$

such that for each $i = 1, \ldots, d$

$$\widetilde{\mathcal{S}}_i = \widetilde{\mathcal{S}}_i^1 \oplus \cdots \oplus \widetilde{\mathcal{S}}_i^{t_i} \tag{4}$$

is the search strategy such that $\widetilde{\mathcal{S}}_i^j = \mathcal{S}_i^j$ for each $j = 1, \ldots, t_i$, while the search strategies (if any) $\widetilde{\mathcal{S}}_i^{t_i+1}, \ldots, \widetilde{\mathcal{S}}_i^{t_i}$ are the ones among $\mathcal{S}_{d+1}, \ldots, \mathcal{S}_l$ that clear subtrees of $T_{\pi(i)}$. Note that in particular $\widetilde{\mathcal{S}}_i^1 = \langle r, \pi(i)\rangle$ for each $i = 1, \ldots, d$. Also, $\widetilde{\mathcal{S}}$ is a valid partial search strategy for T with homebase r, because the order in which the search strategies $\mathcal{S}_{d+1}, \ldots, \mathcal{S}_l$ are performed in $\widetilde{\mathcal{S}}$ is the same as in $\mathcal{S}$. Moreover, $\widetilde{\mathcal{S}}$ is greedy, because $C_E(\widetilde{\mathcal{S}}) = C_E(\mathcal{S})$. Hence, $\widetilde{\mathcal{S}}$ is simple. We say that $\widetilde{\mathcal{S}}$ is the simple search strategy that *corresponds* to $\mathcal{S}$.

As an example let us consider the search strategy $\mathcal{S}$ presented in Fig. 1, where the shaded areas together with the moves that clear the edges in E_r give the search strategies $\mathcal{S}_1 = \langle r, u_1\rangle \oplus \mathcal{S}(T_{u_1}) \oplus \mathcal{S}(T_{u_2})$, $\mathcal{S}_2 = \langle r, u_3\rangle \oplus \mathcal{S}(T_{u_3})$ and $\mathcal{S}_3 = \langle r, u_4\rangle$. The simple search strategy that corresponds to $\mathcal{S}$ is as follows: $\widetilde{\mathcal{S}} = \widetilde{\mathcal{S}}_1 \oplus \widetilde{\mathcal{S}}_2 \oplus \widetilde{\mathcal{S}}_3$, where

$$\begin{aligned}
\widetilde{\mathcal{S}}_1 &= \langle r, u_1\rangle \oplus \mathcal{S}(T_{u_1}) \oplus \mathcal{S}(T_{u_2}) \oplus \mathcal{S}(T_{u_6}) \oplus \mathcal{S}(T_{u_{10}}),\\
\widetilde{\mathcal{S}}_2 &= \langle r, u_3\rangle \oplus \mathcal{S}(T_{u_3}) \oplus \mathcal{S}(T_{u_7}) \oplus \mathcal{S}(T_{u_{11}}),\\
\widetilde{\mathcal{S}}_3 &= \langle r, u_4\rangle \oplus \mathcal{S}(T_{u_4}) \oplus \mathcal{S}(T_{u_5}) \oplus \mathcal{S}(T_{u_8}) \oplus \mathcal{S}(T_{u_9}) \oplus \mathcal{S}(T_{u_{12}}).
\end{aligned}$$

Note that in this example $\mathrm{s}(\widetilde{\mathcal{S}}) > \mathrm{s}(\mathcal{S})$.

Now we prove a technical result that will be used in the proof of the next lemma. Informally speaking, we prove that all vertices $u \in \delta(\mathcal{S}_1, \ldots, \mathcal{S}_p), p \geq d$, such that their weight is 'large', i.e. $w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_p) \setminus \{u\}) < \min\{w(u), w(r)/2\}$, belong to a unique subtree T_v for some $v \in V_r$. Note that $\delta(\mathcal{S}_1, \ldots, \mathcal{S}_p)$ may contain at most one such vertex u, but $\delta(\mathcal{S}_1, \ldots, \mathcal{S}_{p+1})$ may contain another one.

Lemma 9. *Let T be a weighted tree rooted at r and let $\mathcal{S}$ be a greedy search strategy for T of the form as in Theorem 3 such that $\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_j$ is not greedy for each $j = d, \ldots, l-1$. There exists $i \in \{1, \ldots, d\}$ such that for each $p \in \{d, \ldots, l-1\}$ and $u \in \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_p)$ satisfying $w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_p) \setminus \{u\}) < \min\{w(u), w(r)/2\}$ it holds $u \in V(T_{\pi(i)})$.*

Proof. If there exists at most one pair p and u that satisfies the condition in the lemma, then there is nothing to prove. Thus, assume that p_1, u_1 and $p_2, u_2, d \leq p_1 < p_2 < l, u_s \in \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_s}), s = 1, 2$, are two such pairs, i.e.

$$w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_s}) \setminus \{u_s\}) < w(u_s) \text{ and } w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_s}) \setminus \{u_s\}) < w(r)/2, \tag{5}$$

$s = 1, 2$. Suppose for a contradiction that u_s belongs to $V(T_{\pi(j_s)}), s = 1, 2$, and $j_1 \neq j_2$. It holds $w(r) < w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_2}))$, because otherwise, by Lemma 3, $\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_2}$ would be greedy, which would contradict our assumption. Hence,

$$w(r)/2 < w(u_2), \tag{6}$$

because, by (5), $w(r) < w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_2})) = w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_2}) \setminus \{u_2\}) + w(u_2) < 2w(u_2)$. If $u_2 \in \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_1})$, then let $u_2' = u_2$ and otherwise let u_2' be the ancestor of u_2 that belongs to $\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_1})$. Since $u_2' \neq r$ (because $p_1 \geq d$ implies that $r \notin \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_1})$) and $\mathcal{S}_q$ is greedy for each $q = d+1, \ldots, l$, $w(u_2') \geq w(u_2)$. Then, note that $w(u_2') \leq w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_1}) \setminus \{p_1\})$, because $u_2' \in \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_1}) \setminus \{u_1\}$. Thus, $w(u_2) \leq w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_1}) \setminus \{u_1\})$. This, (5) and (6) give us

$$w(r)/2 < w(u_2) \leq w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p_1}) \setminus \{u_1\}) < w(r)/2,$$

which is a contradiction. □

Let $v \in V(T)$ and let u be a child of v in a weighted rooted tree T. Define $\pi_u^v \colon \{1, \ldots, |V_v|\} \to V_v$ to be any permutation of V_v such that $\pi_u^v(|V_v|) = u$ (the remaining vertices in V_v are ordered arbitrarily). Then, let for each set $V_v \in \mathcal{V}$

$$\Gamma_1(V_v) = \{\pi_u^v \colon u \in V_v\}.$$

For the purpose of the proof of the next lemma we introduce the following notation. Given any partial search strategy $\mathcal{S}$ for T and an ordering Γ for T, we use the symbol $\xi_\Gamma(\mathcal{S})$ to denote a partial search Γ-strategy for T such that $C_E(\mathcal{S}) = C_E(\xi_\Gamma(\mathcal{S}))$ and the number of searchers used by $\xi_\Gamma(\mathcal{S})$ is minimum, i.e. $\mathrm{s}(\xi_\Gamma(\mathcal{S})) \leq \mathrm{s}(\mathcal{S}')$ for each partial Γ-search strategy $\mathcal{S}'$ for T such that $C_E(\mathcal{S}') = C_E(\mathcal{S})$. Note that it follows directly from the definition that if $\mathcal{S}$ is greedy, then $\xi_\Gamma(\mathcal{S})$ is greedy as well.

Lemma 10. *Let T be a weighted rooted tree. If there exists a greedy k-search strategy $\mathcal{S}$ for T, then there exists a greedy $3k$-search Γ_1-strategy for T that clears the edges in $C_E(\mathcal{S})$.*

Proof. Let r be the root of T and let π be the permutation that determines the order of clearing the edges in E_r by $\mathcal{S}$. Assume without loss of generality that $\mathcal{S}$ is as in Theorem 3. Let $\widetilde{\mathcal{S}}$ be the simple search strategy that corresponds to $\mathcal{S}$. Recall that $\widetilde{\mathcal{S}}$ has the form as in (3) and in (4).

First we select an integer $a \in \{1, \ldots, d\}$ as follows. If there exist $p \in \{d, \ldots, l-1\}$ and $u \in \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_p)$ that satisfy the condition in Lemma 9, then let $a = i$, where $u \in V(T_{\pi(i)})$, and otherwise let $a = d$. By the definition, there exists a permutation $\pi' \in \Gamma_1(V_r)$ such that $\pi'(d) = \pi(a)$. Let $p_i, i = 1, \ldots, d$, be the integer such that $\widetilde{\mathcal{S}}_{p_i}$ clears $r\pi'(i)$ and a subtree of $T_{\pi'(i)}$. Hence,

$$\widetilde{\mathcal{S}}_{p_1} \oplus \cdots \oplus \widetilde{\mathcal{S}}_{p_d} \tag{7}$$

is a simple search strategy for T that clears the edges in E_r according to the permutation π'. The search strategy in (7) is greedy due to the fact that it clears exactly the edges in $C_E(\mathcal{S})$. Informally speaking, we obtain the latter search strategy by taking the simple search strategy $\widetilde{\mathcal{S}}$ that corresponds to $\mathcal{S}$ and by changing the order in which $\widetilde{\mathcal{S}}$ 'processes' the subtrees T_u, $u \in V_r$, so that the edges in E_r are cleared according to π'. Thus, in particular, the last subtree T_u, $u \in V_r$, partially cleared by the search strategy in (7) is either the same as in $\widetilde{\mathcal{S}}$, or it is the one the contains at least one vertex u from Lemma 9.

Note that the search strategy in (7) may use a different number of searchers than $\widetilde{\mathcal{S}}$. Observe that $\widetilde{\mathcal{S}}^1_{p_i} = \mathcal{S}^1_{p_i} = \langle r, \pi'(i)\rangle$ for each $i = 1, \ldots, d$.

Suppose for a contradiction that $\mathcal{S}$ is a greedy k-search strategy such that there exists no greedy $3k$-search Γ_1-strategy for T that clears the edges in $C_E(\mathcal{S})$. Moreover, assume without loss of generality that $|\mathcal{S}|$ is minimum among all such search strategies $\mathcal{S}$.

Since we work in this proof only with the ordering Γ_1 we will write for brevity ξ in place of ξ_{Γ_1}.

Let us consider the following partial search strategy for T

$$\widetilde{\mathcal{S}}^1_{p_1} \oplus \xi(\widetilde{\mathcal{S}}^2_{p_1}) \oplus \cdots \oplus \xi(\widetilde{\mathcal{S}}^1_{p_1}) \quad \oplus \cdots \oplus \quad \widetilde{\mathcal{S}}^1_{p_d} \oplus \xi(\widetilde{\mathcal{S}}^2_{p_d}) \oplus \cdots \oplus \xi(\widetilde{\mathcal{S}}^{l_d}_{p_d}). \tag{8}$$

Since $\pi' \in \Gamma_1(V_r)$, $\widetilde{\mathcal{S}}^1_{p_i} = \langle r, \pi'(i)\rangle$ and $C_E(\xi(\widetilde{\mathcal{S}}^j_i)) = C_E(\widetilde{\mathcal{S}}^j_i)$ for each $i = 1, \ldots, d, j = 1, \ldots, l_i$, we obtain that the latter is a (simple) greedy search Γ_1-strategy for T that clears the edges in $C_E(\mathcal{S})$. Thus, in order to obtain the desired contradiction it is enough to argue that the search strategy in (8) uses at most $3\mathtt{s}(\mathcal{S})$ searchers.

Our proof uses induction on the length of a search strategy. To give the formal statement of our induction hypothesis we need a lexicographical ordering for ordered pairs of integers. Given four integers a, b, a', b' let $(a, b) \trianglelefteq (a', b')$ if and only if $a < a'$, or $a = a'$ and $b \leq b'$. Let $P = \{(i, j) : i = 1, \ldots, d, j = 2, \ldots, l_i\}$. We prove by induction on $(i, j) \in P$ that the partial search strategy

$$\widetilde{\mathcal{S}}^1_{p_1} \oplus \xi(\widetilde{\mathcal{S}}^2_{p_1}) \oplus \cdots \oplus \xi(\widetilde{\mathcal{S}}^1_{p_1}) \quad \oplus \cdots \oplus \quad \widetilde{\mathcal{S}}^1_{p_{i-1}} \oplus \xi(\widetilde{\mathcal{S}}^2_{p_{i-1}}) \oplus \cdots \oplus \xi(\widetilde{\mathcal{S}}^{l_{i-1}}_{p_{i-1}}) \oplus \widetilde{\mathcal{S}}^1_{p_i} \oplus \xi(\widetilde{\mathcal{S}}^2_{p_i}) \oplus \cdots \oplus \xi(\widetilde{\mathcal{S}}^j_{p_i}) \tag{9}$$

uses at most $3\mathtt{s}(\mathcal{S})$ searchers. Note that this in particular implies that the search strategy in (8) uses at most $3\mathtt{s}(\mathcal{S})$ searchers (by taking $(i, j) = (d, l_d)$). Due to the simplifying assumptions from Section 2 and due to (1) we obtain that we can skip the analysis for the pair $(i, 1)$ for each $i = 1, \ldots, d$.

Suppose that the claim holds for some $(i, j) \in P$, where $(i, j) \neq (d, l_d)$. (We omit the analysis for the base case $(1, 2)$, because it is analogous to the one that follows for the general case.) For brevity, denote the search strategy in (9) by $\overline{\mathcal{S}}$. We argue that the claim holds for $\overline{\mathcal{S}} \oplus \xi(\widetilde{\mathcal{S}}^{j+1}_{p_i})$. (We assume without loss of generality that $j < l_i$, for otherwise the claim needs to be proven for $\overline{\mathcal{S}} \oplus \widetilde{\mathcal{S}}^1_{p_{i+1}} \oplus \xi(\widetilde{\mathcal{S}}^2_{p_{i+1}})$ and the proof is analogous.) Let v be the homebase of $\xi(\widetilde{\mathcal{S}}^{j+1}_{p_i})$. If $\mathtt{s}(\overline{\mathcal{S}}) \geq \mathtt{s}(\overline{\mathcal{S}} \oplus \xi(\widetilde{\mathcal{S}}^{j+1}_{p_i}))$, then the claim follows immediately from the induction hypothesis, so assume that this inequality does not hold. This in particular implies, by Lemma 1, that

$$\mathtt{s}(\overline{\mathcal{S}} \oplus \xi(\widetilde{\mathcal{S}}^{j+1}_{p_i})) = w(\delta(\overline{\mathcal{S}}) \setminus \{v\}) + \mathtt{s}(\xi(\widetilde{\mathcal{S}}^{j+1}_{p_i})). \tag{10}$$

For brevity we use the symbols $\mathcal{S}^*$ and $\mathcal{S}'$ to denote the following search strategies. If $j + 1 \leq t_i$, then let $\mathcal{S}' = \mathcal{S}^{j+1}_{p_i}$, while otherwise let $\mathcal{S}' = \mathcal{S}_p$, where $p \in \{d+1, \ldots, l\}$ and $C_E(\mathcal{S}_p) = C_E(\xi(\widetilde{\mathcal{S}}^{j+1}_{p_i}))$. Hence, in both cases $\mathcal{S}'$ is the part of $\mathcal{S}$ that clears the edges cleared by $\xi(\widetilde{\mathcal{S}}^{j+1}_{p_i})$. Then, let $\mathcal{S}^*$ be selected in such a way that $\mathcal{S}^* \oplus \mathcal{S}'$ can be extended to $\mathcal{S}$. Informally speaking, $\mathcal{S}^*$ is the part of $\mathcal{S}$ prior to $\mathcal{S}'$. Observe that the minimality of $|\mathcal{S}|$ in particular implies that

$$\mathtt{s}(\xi(\widetilde{\mathcal{S}}^{j+1}_i)) \leq 3\mathtt{s}(\mathcal{S}'), \tag{11}$$

because $\mathcal{S}'$ is greedy.

Claim 1. *The following conditions hold:*

(i) $\sum_{q=1}^{i-1} w(V(T_{\pi'(q)}) \cap \delta(\overline{\mathcal{S}})) \leq w(r)$,
(ii) $w(V(T_{\pi'(q)}) \cap \delta(\overline{\mathcal{S}})) \leq w(V(T_{\pi'(q)}) \cap \delta(\mathcal{S}^*))$ *for each* $q \in \{1, \ldots, i-1\}$ *such that* $V(T_{\pi'(q)}) \cap C_V(\mathcal{S}^*) \neq \emptyset$,
(iii) $V(T_{\pi'(i)}) \cap \delta(\overline{\mathcal{S}}) = V(T_{\pi'(i)}) \cap \delta(\mathcal{S}^*)$,
(iv) $V(T_{\pi'(q)}) \cap \delta(\overline{\mathcal{S}}) = \emptyset$ *for each* $q = i+1, \ldots, d$,
(v) $r \in \delta(\overline{\mathcal{S}})$ *if and only if* $i < d$.

Prof of Claim 1. First note that it follows from the definition of $\widetilde{\mathcal{S}}$ that $V(T_{\pi'(q)}) \cap \delta(\overline{\mathcal{S}}) = V(T_{\pi'(q)}) \cap \delta(\mathcal{S})$ for each $q = 1, \ldots, i-1$. Since $\mathcal{S}$ is greedy, $w(\delta(\mathcal{S})) \leq w(r)$. This proves (i). Condition (ii) then follows from the fact that $\mathcal{S}_p$ is greedy for each $p = d+1, \ldots, l$, while (iii) follows from the construction of $\widetilde{\mathcal{S}}$, i.e. from the fact that the search strategies $\mathcal{S}_{d+1}, \ldots, \mathcal{S}_l$ that clear subtrees of $T_{\pi'(i)}$ are 'executed' in the same order both in $\mathcal{S}$ and in $\widetilde{\mathcal{S}}$. Note that

$$\delta(\overline{\mathcal{S}}) = \begin{cases} \bigcup_{q=1}^{i} V(T_{\pi'(q)}) \cap \delta(\overline{\mathcal{S}}) \cup \{r\} & \text{if } i < d \\ \bigcup_{q=1}^{i} V(T_{\pi'(q)}) \cap \delta(\overline{\mathcal{S}}) & \text{if } i = d. \end{cases} \tag{12}$$

This gives (iv) and (v) and ends the proof of Claim 1. □

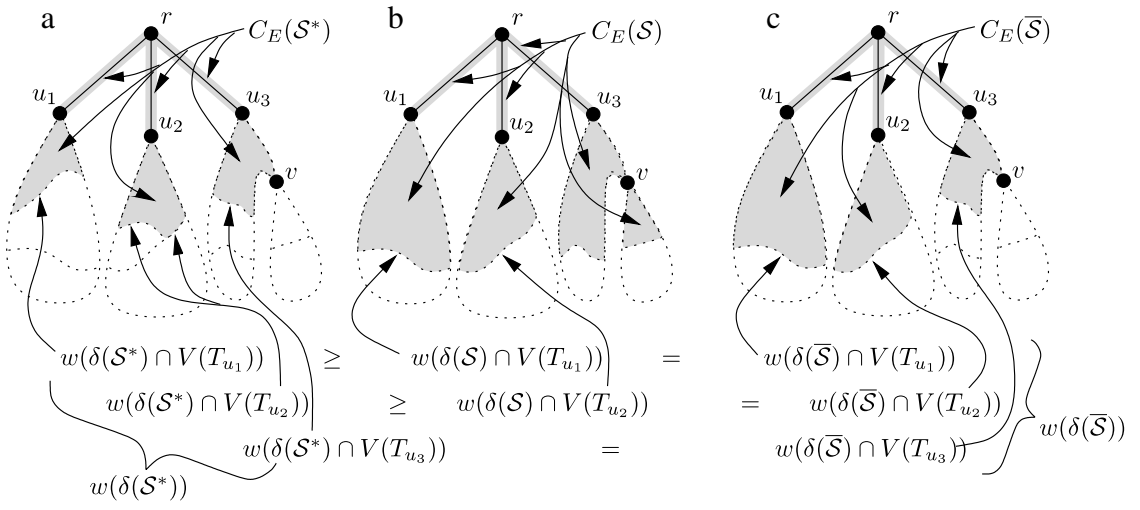

Fig. 2. Case 1: (a) $\mathcal{S}^*$; (b) $\mathcal{S}$; (c) $\overline{\mathcal{S}}$.

In order to use Lemma 9 in the analysis that follows, we need to prove the following fact.

Claim 2. *$\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_q$ is not greedy for each $q = d, \ldots, l-1$.*

Proof of Claim 2. Suppose for a contradiction that $\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_q$ is greedy for some $q \in \{d, \ldots, l-1\}$. We argue by induction on $t = q, \ldots, l$ that

$$\mathtt{s}(\xi(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_t)) \leq 3\mathtt{s}(\mathcal{S}). \tag{13}$$

By the minimality of $|\mathcal{S}|$ and by the fact that $\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_q$ is greedy, $\mathtt{s}(\xi(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_q)) \leq 3\mathtt{s}(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_q) \leq 3\mathtt{s}(\mathcal{S})$ and the claim follows for $t = q$. Thus, assume that the induction hypothesis holds for some $t \in \{q, \ldots, l-1\}$ and we analyze the case for $t+1$. Let us consider $\xi(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_t) \oplus \xi(\mathcal{S}_{t+1})$. By the definition of ξ, the latter is a Γ_1-strategy. We argue that it uses at most $3\mathtt{s}(\mathcal{S})$ searchers. Note that $\delta(\xi(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_t)) = \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_t)$. Let $u \in \delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_t)$ be the vertex such that $\mathcal{S}_{t+1}$ is a partial search strategy for T_u. Then, by the minimality of $|\mathcal{S}|$,

$$\begin{aligned} w(\delta(\xi(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_t)) \setminus \{u\}) + \mathtt{s}(\xi(\mathcal{S}_{t+1})) &\leq w(\delta(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_t) \setminus \{u\}) + 3\mathtt{s}(\mathcal{S}_{t+1}) \\ &\leq 3\mathtt{s}(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{t+1}) \leq 3\mathtt{s}(\mathcal{S}). \end{aligned}$$

This, (13) and Lemma 1 give that $\mathtt{s}(\xi(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{t+1})) \leq \mathtt{s}(\xi(\mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_t) \oplus \xi(\mathcal{S}_{t+1})) \leq 3\mathtt{s}(\mathcal{S})$ as desired. Hence, $\mathtt{s}(\xi(\mathcal{S})) \leq 3\mathtt{s}(\mathcal{S})$, which violates our earlier assumption on $\mathcal{S}$. This completes the proof of Claim 2. □

We consider the following cases.

Case 1: $i = d$. Note that $p_d = a$ and, due to Claim 1(v), $r \notin \delta(\overline{\mathcal{S}})$, i.e. the root r is not guarded during the moves of $\xi(\widetilde{\mathcal{S}}_{p_d}^{j+1})$.

This situation is illustrated in Fig. 2, where $\pi = (u_1, u_2, u_3)$ and $\pi'(d) = u_3$. Hence, $p_d = d$ in this example. Fig. 2(b) gives the border and the subtree of T (the subtree marked with grey) cleared by $\mathcal{S}$. $\mathcal{S}^*$ is given in Fig. 2(a), while Fig. 2(c) depicts the search strategy $\overline{\mathcal{S}}$.

If $p_d = d$, then by Claim 1(ii)–(iv) we obtain that

$$w(\delta(\overline{\mathcal{S}}) \setminus \{v\}) \leq w(\delta(\mathcal{S}^*) \setminus \{v\}).$$

If $p_d \neq d$, then $r \in \delta(\mathcal{S}^*)$ and therefore by Claim 1(i), Claim 1(ii) and Claim 1(iv),

$$w(\delta(\overline{\mathcal{S}}) \setminus \{v\}) \leq w(r) + w(V(T_{\pi'(i)}) \cap \delta(\overline{\mathcal{S}})) \leq w(\delta(\mathcal{S}^*) \setminus \{v\}).$$

Thus, in both cases we obtain by (10) and (11) that

$$\mathtt{s}(\overline{\mathcal{S}} \oplus \xi(\widetilde{\mathcal{S}}_{p_d}^{j+1})) = w(\delta(\overline{\mathcal{S}}) \setminus \{v\}) + \mathtt{s}(\xi(\widetilde{\mathcal{S}}_{p_d}^{j+1})) \leq w(\delta(\mathcal{S}^*) \setminus \{v\})) + 3\mathtt{s}(\mathcal{S}'). \tag{14}$$

By Lemma 1 and by the definitions of $\mathcal{S}^*$ and $\mathcal{S}'$, $w(\delta(\mathcal{S}^*) \setminus \{v\}) + \mathtt{s}(\mathcal{S}') \leq \mathtt{s}(\mathcal{S}^* \oplus \mathcal{S}') \leq \mathtt{s}(\mathcal{S})$. Hence, (14) and $\mathtt{s}(\mathcal{S}') \leq \mathtt{s}(\mathcal{S})$ complete the proof in this case.

Case 2: $i < d$, $p_i \neq d$ and $j+1 \leq t_i$. We have that $\mathcal{S}' = \mathcal{S}_{p_i}^{j+1}$. By Claim 1(v), $r \in \delta(\overline{\mathcal{S}})$. See Fig. 3 that serves as an example in this case, where $\pi = (u_1, u_2, u_3)$, $\pi' = (u_2, u_1, u_3)$ and $i = 2$.

By Claim 1(i), and by (10), (11)

$$\begin{aligned} \mathtt{s}(\overline{\mathcal{S}} \oplus \xi(\widetilde{\mathcal{S}}_{p_i}^{j+1})) &\leq w(r) + \sum_{q=1}^{i-1} w(V(T_{\pi'(q)}) \cap \delta(\overline{\mathcal{S}})) + w(V(T_{\pi'(i)}) \cap \delta(\overline{\mathcal{S}}) \setminus \{v\}) + 3\mathtt{s}(\mathcal{S}') \\ &\leq 2w(r) + w(V(T_{\pi'(i)}) \cap \delta(\overline{\mathcal{S}}) \setminus \{v\}) + 3\mathtt{s}(\mathcal{S}'). \end{aligned} \tag{15}$$

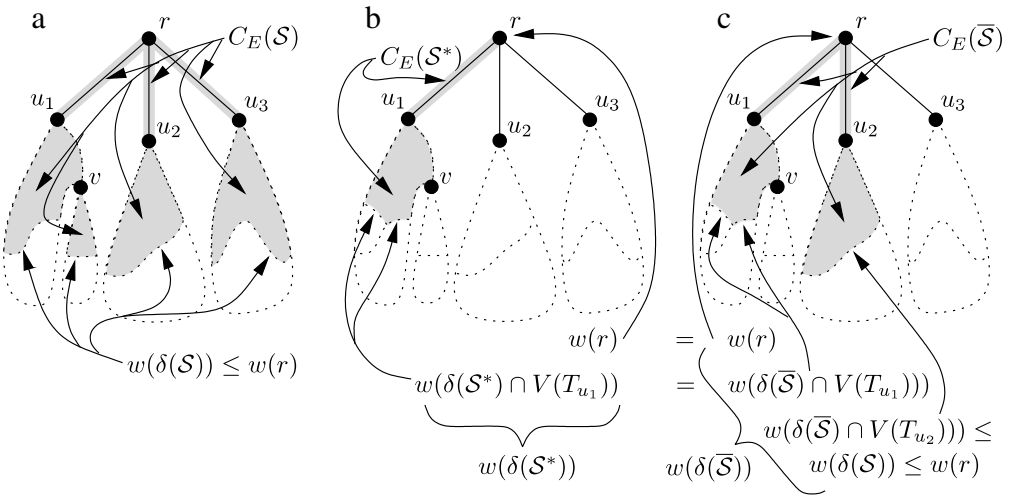

Fig. 3. Case 2: (a) $\mathcal{S}$; (b) $\mathcal{S}^*$; (c) $\overline{\mathcal{S}}$, where $\pi' = (u_2, u_1, u_3)$.

Note that $p_i \neq d$ implies that r is guarded during the moves of $\mathcal{S}' = \mathcal{S}_{p_i}^{j+1}$ in $\mathcal{S}$, or in other words, $r \in \delta(\mathcal{S}^*)$. Thus, by Lemma 1,

$$w(r) + w(V(T_{\pi'(i)}) \cap \delta(\mathcal{S}^*) \setminus \{v\}) + \mathtt{s}(\mathcal{S}') \le w(\delta(\mathcal{S}^*) \setminus \{v\}) + \mathtt{s}(\mathcal{S}') \le \mathtt{s}(\mathcal{S}^* \oplus \mathcal{S}') \le \mathtt{s}(\mathcal{S}).$$

Hence, Claim 1(iii) together with (15) finishes the proof in this case.

Case 3: $i < d$, $p_i \neq d$ and $j + 1 > t_i$. By Claim 2, $\mathcal{S}^*$ is not greedy. Thus, Lemma 3 gives that

$$w(r) \le w(\delta(\mathcal{S}^*)). \tag{16}$$

By (10) by Claim 1(iv) and by Claim 1(v),

$$\begin{aligned}\mathtt{s}(\overline{\mathcal{S}} \oplus \xi(\widetilde{\mathcal{S}}_{p_i}^{j+1})) &= w(\delta(\overline{\mathcal{S}}) \setminus \{v\}) + \mathtt{s}(\xi(\widetilde{\mathcal{S}}_{p_i}^{j+1})) \\ &= w(r) + \sum_{q=1}^{i-1} w(V(T_{\pi'(q)}) \cap \delta(\overline{\mathcal{S}})) + w(V(T_{\pi'(i)}) \cap \delta(\overline{\mathcal{S}}) \setminus \{v\}) + \mathtt{s}(\xi(\widetilde{\mathcal{S}}_{p_i}^{j+1})).\end{aligned} \tag{17}$$

Since $\mathcal{S}^* = \mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{p-1}$ for some $p \in \{d+1, \ldots, l\}$, we obtain that $V_r \subseteq C_V(\mathcal{S}^*)$ and hence by Claim 1(ii) and Claim 1(iii),

$$\sum_{q=1}^{i-1} w(V(T_{\pi'(q)}) \cap \delta(\overline{\mathcal{S}})) + w(V(T_{\pi'(i)}) \cap \delta(\overline{\mathcal{S}}) \setminus \{v\}) \le w(\delta(\mathcal{S}^*) \setminus \{v\}). \tag{18}$$

If

$$w(\delta(\mathcal{S}^*) \setminus \{v\}) \ge w(v), \tag{19}$$

then by (11), (16), (17), (18), by the fact that $w(\delta(\mathcal{S}^*)) = w(\delta(\mathcal{S}^*) \setminus \{v\}) + w(v)$, and by Lemma 1,

$$\begin{aligned}\mathtt{s}(\overline{\mathcal{S}} \oplus \xi(\widetilde{\mathcal{S}}_{p_i}^{j+1})) &\le w(\delta(\mathcal{S}^*)) + w(\delta(\mathcal{S}^*) \setminus \{v\}) + 3\mathtt{s}(\mathcal{S}') \\ &\le 3w(\delta(\mathcal{S}^*) \setminus \{v\}) + 3\mathtt{s}(\mathcal{S}') \le 3\mathtt{s}(\mathcal{S}^* \oplus \mathcal{S}') \le 3\mathtt{s}(\mathcal{S}).\end{aligned} \tag{20}$$

If

$$w(r) \le 2w(\delta(\mathcal{S}^*) \setminus \{v\}), \tag{21}$$

then by (17), (18) and by Lemma 1,

$$\mathtt{s}(\overline{\mathcal{S}} \oplus \xi(\widetilde{\mathcal{S}}_{p_i}^{j+1})) \le 3w(\delta(\mathcal{S}^*) \setminus \{v\}) + 3\mathtt{s}(\mathcal{S}') \le 3\mathtt{s}(\mathcal{S}). \tag{22}$$

The construction of the search strategy in (8), the assumption that $i < d$, Claim 2 and Lemma 9 imply that (19) or (21) holds.

Case 4: $i < d$ and $p_i = d$. Note that from the assumption that $t_d = 1$ it follows that $\mathcal{S}' = \mathcal{S}_p$ for some $p \in \{d+1, \ldots, l\}$. Also, $r \notin \delta(\mathcal{S}^*)$, and by Claim 1(v), $r \in \delta(\overline{\mathcal{S}})$.

First we argue that

$$w(\delta(\mathcal{S}^*) \setminus \{v\}) \ge w(r)/2. \tag{23}$$

Suppose for a contradiction that this does not hold. We have that $v \in \delta(\mathcal{S}^*)$, which implies that $w(v) = w(\delta(\mathcal{S}^*)) - w(\delta(\mathcal{S}^*) \setminus \{v\})$. Since, due to Claim 2, $\mathcal{S}^*$ is not greedy, Lemma 3 gives that $w(\delta(\mathcal{S}^*)) > w(r)$. Thus, $w(v) > w(r) - w(r)/2 = w(r)/2$. Hence, we obtain that $w(\delta(\mathcal{S}^*) \setminus \{v\}) < w(r)/2 < w(v)$. This in particular implies that p and v satisfy the conditions in Lemma 9, which contradicts $i < d$ (in other words, due to the choice of a, we should have $\pi'(d) = \pi(d)$, contrary to $\pi'(i) = \pi(d)$, $i < d$). This gives the desired contradiction and proves (23).

By (23) and by Lemma 1 we obtain

$$w(r)/2 + \mathtt{s}(\mathcal{S}') \leq w(\delta(\mathcal{S}^*) \setminus \{v\}) + \mathtt{s}(\mathcal{S}') \leq \mathtt{s}(\mathcal{S}). \tag{24}$$

Note that $r \notin \delta(\mathcal{S}^*)$ implies that $V(T_{\pi'(q)}) \cap C_V(\mathcal{S}^*) \neq \emptyset$ for each $q = 1, \ldots, d$. Hence, by Claim 1(ii)–(iii), $w(\delta(\overline{\mathcal{S}}) \setminus \{r\})) \leq w(\delta(\mathcal{S}^*))$ and hence

$$w(\delta(\overline{\mathcal{S}}) \setminus \{r, v\}) + \mathtt{s}(\mathcal{S}') \leq w(\delta(\mathcal{S}^*) \setminus \{v\}) + \mathtt{s}(\mathcal{S}') \leq \mathtt{s}(\mathcal{S}). \tag{25}$$

Finally we obtain by (10), (24) and (25) that

$$\mathtt{s}(\overline{\mathcal{S}} \oplus \xi(\widetilde{\mathcal{S}}_{p_i}^{j+1})) = w(\delta(\overline{\mathcal{S}}) \setminus \{v\}) + \mathtt{s}(\xi(\widetilde{\mathcal{S}}_{p_i}^{j+1})) \leq w(\delta(\overline{\mathcal{S}}) \setminus \{v, r\}) + w(r) + 3\mathtt{s}(\mathcal{S}') \leq 3\mathtt{s}(\mathcal{S}). \quad \square$$

Define $\Delta = \max\{|N_T(v)| : v \in V(T)\}$. By the construction of Γ_1 we have that $|\Gamma_1(X)| \leq \Delta$ for each $X \in \mathcal{V}$. Thus, Lemmas 6, 8 and 10 then give the following theorem.

Theorem 4. *There exists a $O(\Delta n^3 \log n)$-time 3-approximation algorithm for connected searching of weighted trees.* $\square$

6. Conclusions

The main contribution of this work is a 3-approximation algorithm for finding connected search strategies for weighted trees. This gives a constant factor approximation polynomial-time algorithm for the pathwidth of weighted trees. The latter problem is also known to be NP-complete [28].

Note that the algorithm `GWTAS` can be used to find an optimal connected search strategy for a weighted tree if we provide the ordering Γ such that $\Gamma(X)$ gives all possible permutations of X for each set $X \in \mathcal{V}$. Thus, the constant γ in Lemma 8 is then bounded by $\Delta!$. It has been proven in [12] that there exists an optimal algorithm for connected searching of weighted trees whose complexity is $O(\Delta! n^3 \log(\Delta! n))$. Thus, by Lemma 6, we can slightly improve the worst case complexity of that algorithm, namely we have obtained the following.

Corollary 1. *Let T be a weighted tree and let $n = |V(T)|$. An optimal connected search strategy for T can be computed in time $O(\Delta! n^3 \log n)$.*

There are some interesting open questions that arise. One of them is: does there exist an efficient algorithm with a smaller approximation ratio? Note that an improved algorithm can be obtained e.g. by proving an analogue of Lemma 10 for an appropriately defined permutation set Γ. On the other hand, one may ask about lower bounds, i.e. what is the constant c such that for each $\epsilon > 0$ there is no $(c - \epsilon)$-approximation algorithm for the connected searching of weighted trees, unless P = NP?

Acknowledgment

This work has been partially supported by MNiSW grant N N206 379337 (2009–2011).

References

[1] L. Barrière, P. Flocchini, F.V. Fomin, P. Fraigniaud, N. Nisse, N. Santoro, D.M. Thilikos, Connected graph searching, Information and Computation (in press).
[2] L. Barrière, P. Flocchini, P. Fraigniaud, N. Santoro, Capture of an intruder by mobile agents, in: SPAA'02: Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures, ACM, New York, NY, USA, 2002, pp. 200–209.
[3] L. Barrière, P. Fraigniaud, N. Santoro, D.M. Thilikos, Searching is not jumping, in: WG'03: Proceedings of the 29th International Workshop on Graph-Theoretic Concepts in Computer Science, 2003, pp. 34–45.
[4] D. Bienstock, Graph searching, path-width, tree-width and related problems (a survey), DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 5 (1991) 33–49.
[5] H.L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth, Theoret. Comput. Sci. 209 (1998) 1–45.
[6] H.L. Bodlaender, F.V. Fomin, Approximation of pathwidth of outerplanar graphs, J. Algorithms 43 (2002) 190–200.
[7] H.L. Bodlaender, J.R. Gilbert, H. Hafsteinsson, T. Kloks, Approximating treewidth, pathwidth, frontsize, and shortest elimination tree, J. Algorithms 18 (1995) 238–255.
[8] H.L. Bodlaender, T. Kloks, Efficient and constructive algorithms for the pathwidth and treewidth of graphs, J. Algorithms 21 (1996) 358–402.
[9] H.L. Bodlaender, T. Kloks, D. Kratsch, Treewidth and pathwidth of permutation graphs, SIAM J. Discret. Math. 8 (1995) 606–616.
[10] H. Chou, M. Ko, C. Ho, G. Chen, Node-searching problem on block graphs, Discrete Appl. Math. 156 (2008) 55–75.
[11] H.H. Chou, M.T. Ko, C.W. Ho, G.H. Chen, Node-searching problem on block graphs, Discrete Appl. Math. 156 (1) (2008) 55–75.
[12] D. Dereniowski, Connected searching of weighted trees, Theoret. Comput. Sci. 412 (2011) 5700–5713.
[13] D. Dereniowski, From pathwidth to connected pathwidth, in: STACS, 2011, pp. 416–427.
[14] J. Ellis, M. Markov, Computing the vertex separation of unicyclic graphs, Inform. and Comput. 192 (2004) 123–161.
[15] J.A. Ellis, I.H. Sudborough, J.S. Turner, The vertex separation and search number of a graph, Inform. and Comput. 113 (1994) 50–79.
[16] U. Feige, M.T. Hajiaghayi, J.R. Lee, Improved approximation algorithms for minimum-weight vertex separators, in: STOC, 2005, pp. 563–572.

[17] F.V. Fomin, P. Fraigniaud, D.M. Thilikos, The price of connectedness in expansions, Technical report, Technical Report, UPC Barcelona, 2004.
[18] F.V. Fomin, D.M. Thilikos, A 3-approximation for the pathwidth of Halin graphs, J. Discrete Algorithms 4 (2006) 499–510.
[19] F.V. Fomin, D.M. Thilikos, An annotated bibliography on guaranteed graph searching, Theoret. Comput. Sci. 399 (3) (2008) 236–245.
[20] J. Gustedt, On the pathwidth of chordal graphs, Discrete Appl. Math. 45 (3) (1993) 233–248.
[21] P. Hlinený, S. Oum, D. Seese, G. Gottlob, Width parameters beyond tree-width and their applications, Comput. J. 51 (3) (2008) 326–362.
[22] N.G. Kinnersley, The vertex separation number of a graph equals its path-width, Inform. Process. Lett. 42 (6) (1992) 345–350.
[23] L.M. Kirousis, C.H. Papadimitriou, Interval graphs and searching, Discrete Appl. Math. 55 (1985) 181–184.
[24] L.M. Kirousis, C.H. Papadimitriou, Searching and pebbling, Theoret. Comput. Sci. 47 (2) (1986) 205–218.
[25] T. Kloks, H.L. Bodlaender, H. Müller, D. Kratsch, Computing treewidth and minimum fill-in: All you need are the minimal separators, in: Proceedings of the First Annual European Symposium on Algorithms, Springer-Verlag, London, UK, 1993, pp. 260–271.
[26] A.S. LaPaugh, Recontamination does not help to search a graph, J. ACM 40 (2) (1993) 224–245.
[27] N. Megiddo, S.L. Hakimi, M.R. Garey, D.S. Johnson, C.H. Papadimitriou, The complexity of searching a graph, J. ACM 35 (1) (1988) 18–44.
[28] R. Mihai, I. Todinca, Pathwidth is NP-hard for weighted trees, in: FAW'09: Proceedings of the 3d International Workshop on Frontiers in Algorithmics, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 181–195.
[29] R. Möhring, Graph problems related to gate matrix layout and PLA folding, in: E. Mayr, H. Noltemeier, M. Syslo (Eds.), Computational Graph Theory, in: Computing Supplementum, vol. 7, 1990, pp. 17–51.
[30] N. Nisse, Connected graph searching in chordal graphs, Discrete Appl. Math. 157 (12) (2008) 2603–2610.
[31] N. Robertson, P.D. Seymour, Graph minors: X. obstructions to tree-decomposition, J. Combin. Theory Ser. B 52 (1991) 153–190.
[32] K. Suchan, I. Todinca, Pathwidth of circular-arc graphs, in: WG, 2007, pp. 258–269.
[33] B. Yang, D. Dyer, B. Alspach, Sweeping graphs with large clique number, Discrete Math. 309 (18) (2009) 5770–5780.